CHIP PLUS

EINE BEILAGE
IN CHIP NR. 1
JANUAR 1989

RISC

CHIP WISSEN

Die kompetente Buchreihe rund um den PC



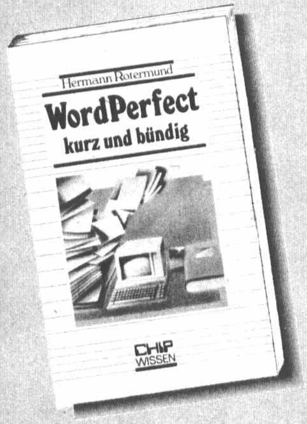
Hans-Peter Förster/Martin Zwernemann:

Word 4.0 kurz und bündig

256 Seiten, 67 Bilder, Hardcover
43,– DM/ISBN 3-8023-**0215**-X

Dieses Arbeitsbuch richtet sich an interessierte Einsteiger sowie an Word-4.0-Anwender, die schnell den Umgang mit dem komfortablen Textverarbeitungsprogramm erlernen wollen. Es ist so aufbereitet, daß es auch als Begleitmaterial für Word-4.0-Schulungen geeignet ist. Aber auch zum Selbststudium ist es in Ergänzung zum Word-4.0-Handbuch ein Hilfsmittel, das viele Tips und Tricks verrät.

- Wichtige Grundfunktionen
- Texte erstellen, korrigieren, speichern
- Texte laden, korrigieren, gestalten
- Texte direkt/indirekt formatieren
- Ausdruck
- Textbausteine
- Rechtschreibprogramm u.a.



Hermann Rotermund:

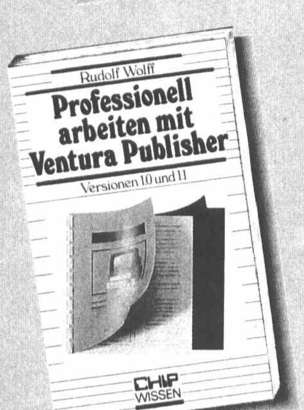
WordPerfect kurz und bündig

144 Seiten, zahlreiche Bilder, Hardcover
30,– DM/ISBN 3-8023-**0121**-8

In diesem Buch werden zunächst die grundlegenden Arbeitsabläufe vorgestellt, dann die jeweils notwendigen Arbeitsschritte für das Erreichen eines bestimmten Ziels erklärt. Außerdem enthält es Anleitungen zur professionellen Nutzung von WordPerfect für spezielle Aufgaben. Die Erläuterung der Funktionen folgt dem logischen Aufbau des Programms. Dem Buch liegt die deutsche Version 4.2 zugrunde.

Besonderes aus dem Inhalt:

- Makros
- Mischfunktionen
- Rechnen im Text
- Fußnoten und Endnoten
- Inhaltsverzeichnis erstellen, Index und Listen
- Spaltenverarbeitung mit WordPerfect



Rudolf Wolff:

Professionell arbeiten mit Ventura Publisher

264 Seiten, zahlr. Bilder, Hardcover
43,– DM/ISBN 3-8023-**0213**-3

Professionell arbeiten mit Ventura Publisher ist sowohl eine Einführung in die Programmvielfalt des Xerox Ventura Publisher, als auch und gerade ein Praxishandbuch, das dem Leser immer dann hilfreich zur Seite steht, wenn Gestaltungs- und Typografieprobleme auftreten. Nach der Einleitung über Voraussetzungen und Möglichkeiten des Ventura-Publisher und einigen Ausführungen zu Satz-, Layout- und Typografiegrundsätzen folgt die ausführliche Einweisung in das Programm, um die angebotenen Möglichkeiten sich und rationell zu nutzen. Der Praxisteil soll mit Übungen und Beispielen die bis dahin erworbenen theoretischen Kenntnisse vertiefen und dem Anwender Sicherheit vermitteln.



Wolfgang Franzen:

Planung und Budgetierung mit Javelin

280 Seiten, 141 Bilder, Hardcover
48,– DM/ISBN 3-8023-**0206**-0

Den Besitzern von Javelin hilft dieses Buch bei der praktischen Anwendung. Es wurde für alle geschrieben, die ein erprobtes Modell für die Erfolgs- und Liquiditätsplanung und praktische Tips für den erfolgreichen Einsatz von Javelin suchen. Eine Vielzahl von Bildern verdeutlichen die Funktionsweise des Programmpakets. Der Leser kann so jederzeit einzelne Schritte auf dem eigenen PC nachvollziehen.
Eine Diskette mit den im Buch vorgestellten Modellen ist erhältlich.

Haben Sie schon den neuen „CHIP- Katalog"?
Bestellen Sie gleich!

VOGEL Buchverlag Würzburg
Postfach 6740 · 8700 Würzburg 1

Abspecken heißt die Devise

Ulrich Parthier

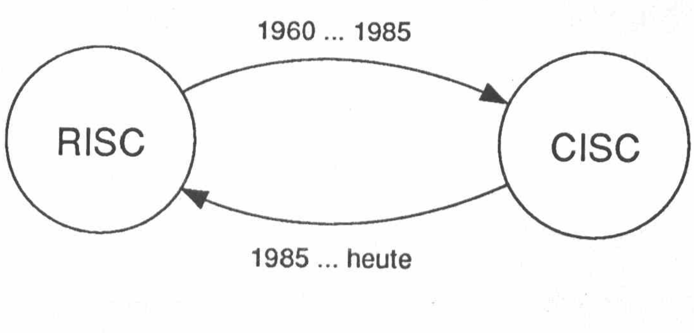
RISC bedeutet nicht, wie viele bei der Geburt dieses Schlagwortes vielleicht meinten, „Risiko", RISC ist auch kein neues Gesellschaftsspiel, etwa eine Variante von Monopoly. Sondern RISC ist eine Abkürzung, hinter der sich der Begriff „Reduced Instruction Set Computer" verbirgt, in Deutsch: ein verkürzter Befehlssatz für Prozessoren.

Bis Mitte der achtziger Jahre arbeiteten die Prozessoren unserer Rechner allein mit sogenannten CISC-Befehlssätzen. CISC steht für „Complex Instruction Set Computer" und spiegelte die Entwicklung zu immer komplexeren und größeren Befehlssätzen wider.

Schon Mitte der siebziger Jahre begann man, sich bei IBM, aber auch an den amerikanischen Eliteuniversitäten in Stanford und Berkeley Gedanken zu machen, ob dies eigentlich sinnvoll sei. Die Antwort war nein. Das überraschende Ergebnis der Forschungen lautete, daß nur 20 Prozent aller Befehle häufig genutzt wurden. Die übrigen 80 Prozent der Befehle trugen nur wenig zur Systemleistung bei. Die Konsequenz dieser Tatsache hieß: Reduktion des Befehlssatzes und der Befehlsformate. „Schlank und rank" lautet heute die Devise.

Während etwa die RISC-Prozessoren von Hewlett-Packard und IBM nur für eigene Systeme Verwendung finden und anderen Computer-Herstellern nicht zur Verfügung stehen, entbrennt auf dem Halbleitermarkt ein Kampf um die Vormachtstellung. Dabei haben die großen Hersteller Intel und Motorola die Entwicklung wohl ein wenig verschlafen. Erst im Frühjahr 1988 kamen von ihnen entsprechende Ankündigungen.

Einen Entwicklungsvorsprung von zwei bis drei Jahren und damit gute Karten haben dagegen Sun Microsystems (mit AT&T als Großaktionär im Hintergrund) mit seiner SPARC-Architektur und das kalifornische Unternehmen Mips Computer (hier hat Digital Equipment ein größeres Aktienpaket übernommen) mit seinem Mips-Prozessor. Sie werden, so scheint es, auch gegen die Riesen des Halbleitergeschäftes bestehen können. Denn bis die von den Mitbewerbern angekündigten Chips lieferbar sind, haben diese beiden schon längst einen schwerwiegenden Vorteil: nämlich eine Vielzahl fertiger Applikationen – und das überzeugt die Entwickler allemal.



RISC: Prädestiniert für ASIC

Uri Mayer, Ulrich Parthier

Die Autoren:
Uri Mayer, Dipl.-Ing. (TU), ist als Marketing Manager Advanced Products bei der LSI Logic GmbH in München beschäftigt.

Ulrich Parthier studierte in Göttingen Deutsch, Sozialkunde, Geschichte und Sport, bevor er über die Nixdorf Computer AG in Paderborn nach München kam. Hier ist er seit zwei Jahren als freier Journalist tätig.

Verschiedene Studien, zum Beispiel die von Patterson und Sequin vom Berkeley-RISC-Projekt, haben HPS-Befehlsarten, Operand-Formen und Adressierungs-Modi in der Häufigkeit ihres Auftretens, des Speicherzugriffs, der Art und des Zeitaufwands pro Befehl in den unterschiedlichsten Programmen untersucht.

Mit dem Ergebnis, daß eine einfache Wortzuweisung vom Typ „A = B" der wichtigste, am häufigsten auftretende Befehl ist, gefolgt von „IF", „CALL" und „LOOP", wobei diese Befehle in der Maschinensprache als „COMPARE", „BRANCH" und „JUMP realisiert werden. Aus den Untersuchungen geht ebenso hervor, daß die Skalar-Variable der am meisten genutzte Operand ist; danach kommen „ARRAY" und „STRUCTURES", die wir aus Pascal oder C kennen. Interessant ist weiter, daß diese Variablen fast nur lokal genutzt und im „Procedure Call" meist weniger als sechs Variablen ausgetauscht werden.

Das alles muß in Betracht gezogen werden, um die richtigen Befehle in einen RISC-Prozessor wie den R3000 von Mips Computer zu implementieren und eine optimale Ausführungsgeschwindigkeit zu erzielen. Ein entsprechender Befehlssatz muß grundsätzlich folgende Kritierien berücksichtigen:

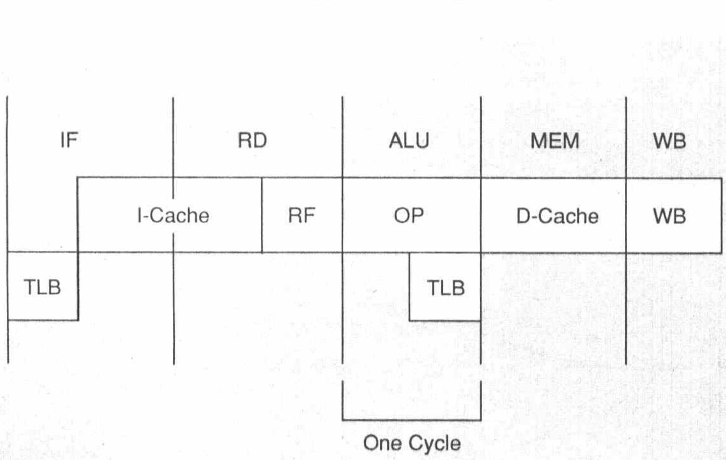
- möglichst alle Instruktionen in nur einem Maschinenzyklus,
- Register-to-Register-Operationen,
- einfaches Instruktionsformat,
- einheitliches Datenformat,
- Koprozessor-Hilfsbefehle.

Der reduzierte Befehlssatz (insgesamt 74) des R3000 kann in sechs Gruppen unterteilt werden:

- LOAD/STORE-Instruktionen, um Daten zwischen Memory und internen Registern zu bewegen. Diese Befehle sind vom Typ „Immediate".
- COMPUTATIONAL – arithmetische, logische und Shift-Befehle (vom Typ „Register oder Immediate").
- JUMP/BRANCH-Instruktionen, um den Programmfluß zu verändern oder Subroutinen aufzurufen. Diese Befehle sind vom Typ „Register, Immediate oder Jump".
- Koprozessor-Instruktionen, um Hilfsprozessoren zu unterstützen (Befehle vom Typ „Immediate").
- Koprozessor-O-Instruktionen, um den On-Chip-Hilfsprozessor, der Memory-Management-Funktion hat, zu unterstützen (Befehle vom Typ „Immediate").
- SPECIAL-Instruktionen zur Systemunterstützung (Typ „Immediate").

Das Instruktionsformat ist also ein weiterer wichtiger Faktor zur Auswahl der Befehle, das auch zur Zeit/Instruktions-Reduzierung beisteuern kann. Angestrebtes Ziel ist die Realisierung eines gleichmäßigen Formats für alle Befehle. Dies führt zur einfachen Instruktions-Decodierung und ermöglicht auch Simultanverarbeitung von Befehlen. Der Befehlssatz des R3000 benutzt drei gleichmäßig 32 bit lange Formate mit konstanten Bereichen. Die komplizierteren, nicht so häufig vorkommenden Befehle und Adressierungsarten werden bei einem RISC-Prozessor mit Hilfe von Compilern auf die einfacheren RISC-Instruktionen abgebildet.

Eine Instruktion wird in der Pipeline in fünf Stufen decodiert und aufgeführt



Die Architektur der Zentraleinheit

Eng verbunden mit dem Instruktionssatz ist die RISC-CPU-Architektur. Auf der Suche nach der optimalen Struktur werden von der Forschung drei Schwerpunkte herausgestellt:

- zeitgleiche Befehle,
- interne Register,
- Instruktions-Pipeline.

Die reduzierten und in Hardware realisierten Befehle, einschließlich Multiplikation und Division, führen zu einer CPU-Architektur, die keinen oder nur einen sehr kleinen Mikrocode benötigt, um sehr schnelle und zeitgleiche Befehle mit möglichst einem Maschinenzyklus zu erreichen. Ebenfalls eine notwendige Voraussetzung für die Instruktions-Pipeline der RISC-Architektur ist die Gleichmäßigkeit von Befehlen.

Die internen Register eines Prozessors sind die schnellsten Speicherelemente. Indem man bei der Auswahl der Befehle ausschließlich Register-to-Register-Operationen nutzen kann, reduziert sich der Zeitaufwand pro Instruktion auf ein Minimum. Die Ergebnisse der verschiedenen Untersuchungen divergieren, wenn es um die Anzahl der zu implementierenden internen Register in einem RISC-Prozessor geht. Es ist meist ein Kompromiß zwischen Hard- und Software. Die eine These vertritt die „Mehr-Register"-Lösung, die andere plädiert für Mehraufwand bei der Entwicklung der Compiler.

Aus verschiedenen Untersuchungen geht außerdem hervor, daß lokale Skalar-Variable die am häufigsten benutzten sind und daß beim „Procedure-Call" wenige Variable (meist noch nicht einmal sechs) ausgetauscht werden. Die optimale Lösung beinhaltet daher ein Minimum an zeitaufwendigen externen Speicherzugriffen, dazu intelligente Compiler-Programme, so daß die im Chip verfügbaren Register maximal mit den in einer Zeitperiode am meisten benutzten Variablen gefüllt werden. Die Arbeitsregister sollen gleichzeitig als „General-Purpose"-Register realisiert werden und können damit als Operand-, Adressen-, Index-Stack-Pointer-Register (und so weiter) dienen.

Ein weiterer Schritt zur Erhöhung der Befehls-Durchsatzrate ist das Instruktions-Pipeline-Prinzip. Außer den zuvor erwähnten zeitgleichen Befehlen ist es notwendig, die einzelnen Instruktionen in kleinere Befehlsabschnitte zu unterteilen. Damit können über Pipelines mehrere Instruktionen simultan bearbeitet werden.

Die fünf typischen Befehlsstufen sind:

- IF: „Instruction Fetch" (Instruction-Cache), TLB = Translation-Look-aside-Buffer)
- RD: „Read" (RF = Register Fetch)
- ALU: „Operation Perform"
- MEM: „Access Memory" (Data Cache)
- WB: „Write Back"

Jeder Teil der Instruktion ist ungefähr einen Taktzyklus lang, wobei Teile, die mehr Zeit beanspruchen, benachbart oder überlappend sind zu anderen, die weniger als eine Taktperiode benötigen. Dieser regelmäßige Aufbau der Instruktionen ermöglicht optimalen Instruktionsfluß. Voraussetzung dafür ist, daß die Pipeline-Stufen immer voll beansprucht werden. Die Verantwortung hierfür liegt bei der Compiler-Technologie.

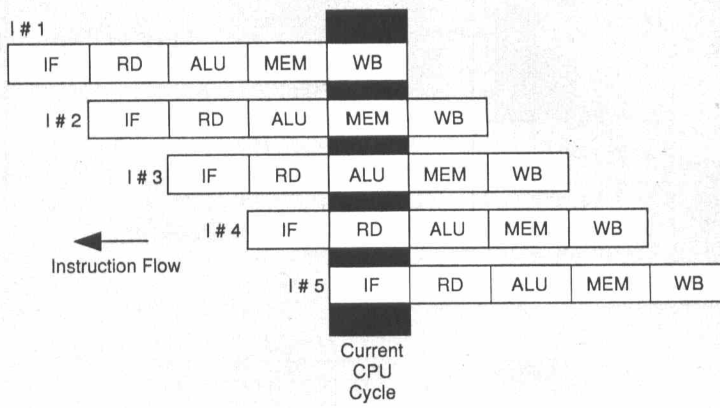
System-Designer notwendig

Die Implementierung der RISC-Prozessoren in ein System garantiert nicht allein für Leistungssteigerung. Der System-Designer ist hier – wie auch bei anderen Systemrealisierungen – gefordert.

Bei den meisten Systemen ist keine Differenzierung zwischen Instruktionen, Datenspeicherung und Art der Adressierung von I/O-Devices erkennbar. Diese Speicherzuordnung führt zu einer Rechnerleistungs-Reduzierung. Schnelle Cache-Speicher werden heute eingesetzt, um häufig benutzte Befehle beziehungsweise Daten (zum Beispiel die Instruktionen innerhalb der Schleife) zu speichern, so daß ein Zugriff auf den langsameren Hauptspeicher seltener erfolgen muß.

Manche der CISC-Mikroprozessoren nutzen diese Technologie ebenfalls und verfügen über einen begrenzten On-Chip-Cache. Der R3000 unterstützt einen großen externen Instruktions- und Daten-Cache-Bereich. Dies führt zu signifikanten Verbesserungen für eine breite Palette von Programmen. Da eine Cache die Eigenschaft besitzt, Daten nur temporär zu speichern, müssen zusätzlich alle Daten im Hauptspeicher gesichert werden. Um auch hier keine kostbare Zeit zu verlieren, sorgt ein Write-Buffer IC aus der gleichen

Im optimalen Fall werden die fünf Pipeline-Stufen immer voll ausgenutzt



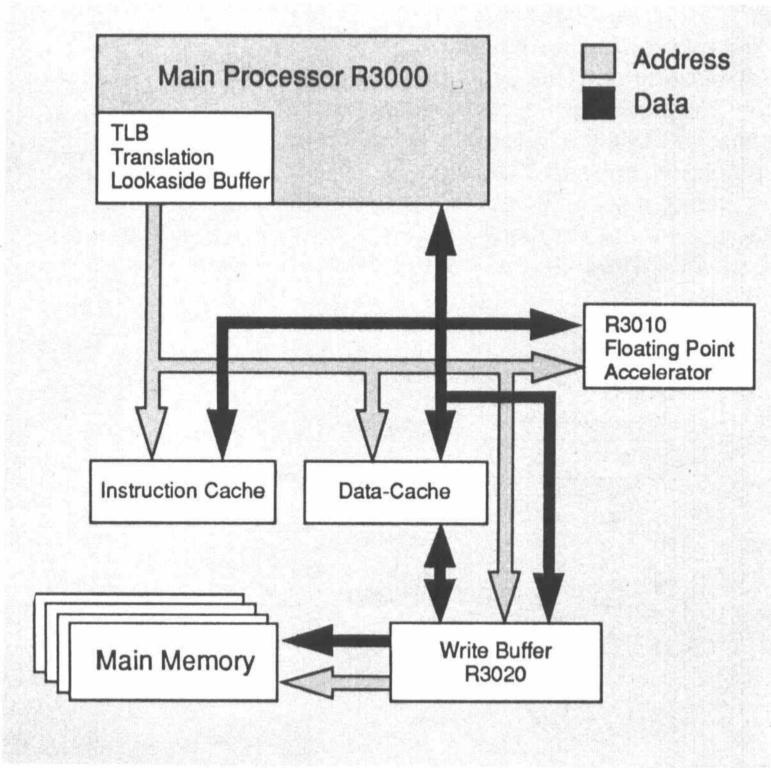
Familie (R3020) für eine Schnittstelle zwischen Prozessor und Hauptspeicher. Der R3020 (viermal pro System vorhanden) erhält – ebenso wie der Cache – die Adressen und Daten und sorgt für den Datentransfer zum Hauptspeicher. Nicht minder wichtig sind die Koprozessoren, die anteilig Aufgaben der zentralen CPU übernehmen.

Hochentwickelter Compiler

Ein RISC-Prozessor ist, wie auch der CISC-Prozessor, mittels Maschinensprache (Assembler) programmierbar. Das Ziel ist jedoch eine Programmierung mit einer geeigneten höheren Programmiersprache (HPS). Umsetzen (Kompilieren) von HPS-Befehlen in Maschineninstruktionen geht in der Regel einfach. Für die Programmierung von RISC-Prozessoren benötigt man einen hochentwickelten Compiler, der interne Register, Pipeline-Strukturen, Spezialbefehle und Koprozessoren unterstützt. Diese Umstände führen dazu, daß der Compiler und die Hardware-Architektur gemeinsam entwickelt werden müssen, wenn sie ein Maximum an Leistung erbringen sollen.

Ein RISC-Prozessor verfügt über eine Vielzahl von internen Arbeitsregistern – beim R3000 sind es 32. Der Compiler versucht nun, so viele aktuelle Variablen wie möglich in den Registern zu speichern, um Zugriffe in den Hauptspeicher zu vermeiden. Eine Studie von A. Tannenbaum (1978) hat ergeben, daß ein laufender Prozeß im Durchschnitt sechs Parameter an einen untergeordneten Prozeß weitergibt: Diese werden vom Compiler so in Registern angeordnet, daß der neue Prozeß sie direkt übernehmen kann. Damit wird die Übergabe von Parametern über einen „Stack" überflüssig. Nur im Fall einer Verschachtelung von untergeordneten Prozessen ist es notwendig, Registerdaten im Cache auszulagern.

Der R3000 unterstützt einen externen Instruktions- und Daten-Code. Die Schnittstelle zwischen Code und Hauptspeicher wird mit dem Write-Buffer R3020 realisiert.



Weitere Aufgaben

Aufgabe des Compilers ist es auch, die Pipelines optimal auszulasten und Spezialbefehle wie „LOAD" und „STORE", die auf den Hauptspeicher zurückgreifen, oder „JUMP" und „BRANCH", die die Zieladressen aus einem Speicher holen müssen, zu unterstützen. Hier ist der Compiler in der Lage, den Befehlsfluß so zu organisieren, daß der Ablauf möglichst reibungslos erfolgt, indem er auf Befehle, die Speicherzugriff benötigen, Instruktionen folgen läßt, die nicht vom externen Speicher abhängig sind. Ähnlich wird bei Multiplikationen, Divisionen und FP-Manipulationen im Koprozessor verfahren, die mehr Maschinenzyklen benötigen. So kann es auch passieren, daß sogenannte „NO-OP"-Befehle generiert werden.

RISC und ASIC

Die Vorteile der RISC-Design-Technik liegen nicht nur in der Rechnerleistungs-Steigerung, sondern ebenso in der Einfachheit der Architektur, die leicht zu implementieren ist und deshalb noch in der Design-Phase schnelle Fehlerbehebung ermöglicht. Im Vergleich dazu erfordern die CISC-Prozessoren, die in der „Full-Custom"-Technik entwickelt werden, lange Entwicklungsphasen. Oft ist die VLSI-Technologie, in der sie realisiert werden, bereits veraltet, bevor sie fehlerfrei funktionieren und auf den Markt gebracht werden können.

Gerade wegen der geringeren Komplexität der Architektur ist der RISC-Prozessor zur Implementierung in ASIC prädestiniert. Stets kann in der modernsten zur Verfügung stehenden VLSI-Technologie entwickelt werden, da ASIC-Methodik immer mit der innovativsten Technologie arbeitet und in der Mikroelektronik-Welt derzeit und wohl auch für die nähere Zukunft die treibende Kraft darstellt.

Skalierbare Prozessorarchitektur

Georg Pöll

Sun Microsystems hat eine RISC-Architektur mit dem Namen SPARC entwickelt. Der Ausdruck SPARC steht für „Skalierbare Prozessorarchitektur“. Die Design-Spezifikationen sind veröffentlicht, und aufgrund dieser Tatsache stellen verschiedene Mikroprozessor-Anbieter Produkte basierend auf dem SPARC-Design her.

Mit dem Ausdruck „skalierbar“ ist die Möglichkeit der Verschmälerung der Leiterbahnen auf dem Chip gemeint. Die Schnelligkeit von Chips steht in direktem Zusammenhang mit der Breite der Leiterbahnen. Trotzdem ist es bei einigen Designs nicht möglich, die Chips kleiner zu machen, also höher zu integrieren, weil die Architektur zu kompliziert ist. Wegen der Einfachheit des Designs bei SPARC kann eine Verkleinerung problemlos erfolgen.

Der Zeitverbrauch eines Programms resultiert nicht nur aus der Ausführungszeit von Instruktionen, sondern auch aus dem Zeitverlust, bedingt durch I/O, sowie dem Hin- und Herschalten zwischen verschiedenen Prozessen. Die Zeitspanne, die sich aus diesen und anderen Overheads ergibt, kann die Prozessorleistung erheblich beeinträchtigen. Ein beliebig schneller Prozessor etwa vermag ein Programm, das 33 Prozent der Gesamtlaufzeit mit Warten auf Ein- und Ausgabe (I/O) verbringt, nur um den Faktor drei in der Laufzeit zu verbessern. Von Systemen, die auf RISC-Architekturen basieren, verspricht man sich einen Ausgleich von Rechen- und der Durchsatzleistung bei der Ein- und Ausgabe.

Die Architektur

Ein auf SPARC basierender Prozessor besteht aus einer Integer-Unit (IU), einer Floating-Point-Unit (FPU) und optional aus einem konstruktionsabhängigen Koprozessor (CP), alle mit eigenen Registern ausgestattet. Diese Struktur erlaubt ein optimales Zusammenwirken von Integer-, Floating-Point und Koprozessor-Instruktionen. Sämtliche Register sind 32 bit lang; abweichen von dieser Länge kann der Koprozessor mit seinen Registern. Instruktionen arbeiten auf einzelnen Registern, auf Registerpaaren oder auf Vierergruppen von Registern.

Im folgenden werden die Integer-, Floating-Point- und Koprozessor-Architekturen mit einer Übersicht aller Instruktionen beschrieben.

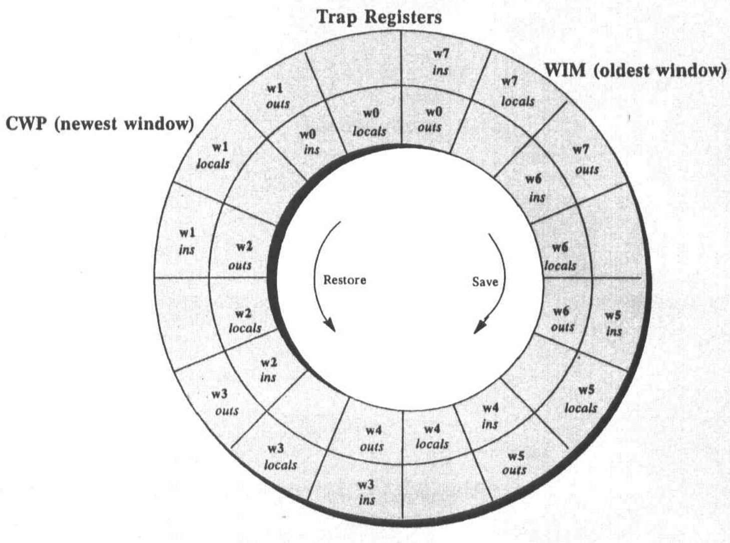
Architektur der Integer-Unit

Die Integer-Unit (IU) kann zwischen 40 und 520 allgemeine 32 bit lange Register enthalten. Diese Zahl ergibt sich aus der Gruppierung von Registern in zwei bis 32 sich überlappende Registerfenster. Die aktuelle Anzahl der tatsächlichen Registerfenster auf einem Chip ist einem Compiler oder einer Applikation nicht bekannt. (Eine Implementierung mit zwei Registerfenstern oder mit 40 Registern funktioniert genauso wie eine Implementierung ohne Registerfenster, nur eben langsamer.)

Ein Programm kann zugleich 32 Integer-Register adressieren: 8 *ins*, 8 *locals*, 8 *outs* des aktiven Registerfensters und 8 *globals*, adressierbar von einem beliebigen Registerfenster. Die 8 *outs* eines Fensters sind identisch mit den 8 *ins* des nächsten Fensters. Obwohl eine Instruktion 24 in ein Fenster eingeteilte Register



Der Autor
Georg Pöll, arbeitete nach einem Studium an der Fachhochschule in München zunächst bei einem herstellerunabhängigen Unix-Software-Haus als Applikationsentwickler. Seit Juli 1987 ist er bei Sun Microsystems in München in der Vertriebsunterstützung tätig.

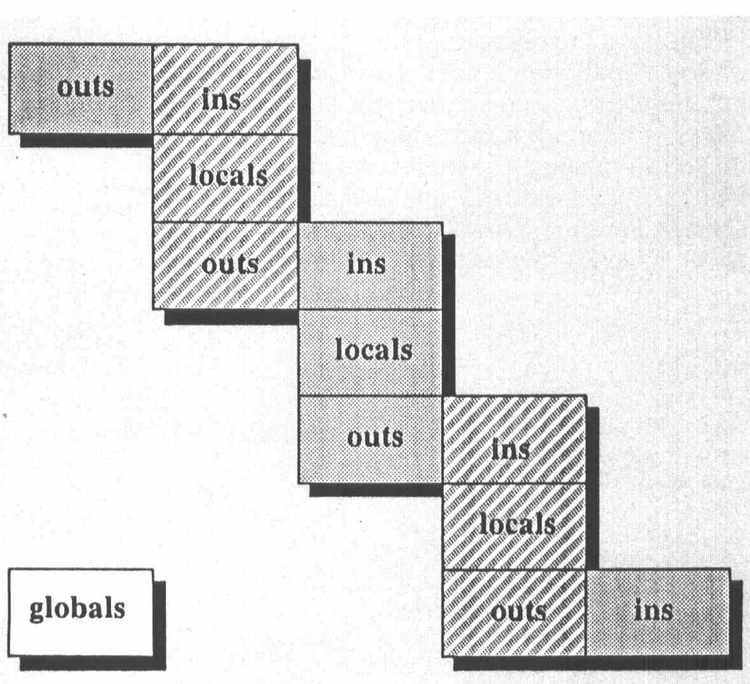


Die Darstellung zeigt eine Implementierung mit acht Registerfenstern

adressieren kann, besteht ein Registerfenster nur aus 16 Registern, 8 *ins* und 8 *locals*. Das globale Register 0 hat dabei immer den Wert 0, womit eine einfache und permanente Nutzung dieses besonderen Registers ermöglicht wird.

Ein sogenannter „Current Window Pointer" (CWP) im Statusregister des Prozessors (PSR), der für den Benutzer nicht sichtbar ist, zeigt auf das aktuelle Registerfenster. Mit der save-Instruktion wird der CWP dekrementiert, wodurch das nächste Fenster zum aktiven Fenster wird. Und weil sich die Fenster überlappen, werden die outs des alten Fensters adressierbar als ins des neuen. Wird der CWP mit der restore-Instruktion inkrementiert, so wird das vorherige zum aktiven Fenster. Typischerweise wird eine save-Instruktion während eines Procedure-Calls ausgeführt und ein restore beim Procedure-Return. In einigen Fällen, wie beim Verlassen von Prozeduren oder bei der Optimierung von „Interprocedural Register Allocation" (IRA), ist es möglich, das Allokieren von neuen Registerfenstern beim Prozeduraufruf zu verhindern. Save und restore führen auch eine Addition von

SPARC-Registermodell. Jede Box besteht aus acht allgemeinen Integer-Registern. Die vertikale Zusammenfassung von ins und locals ergibt ein Fenster. Die Anzahl der Fenster ist implementierungsabhängig.



Register zu Register durch und verändern sowohl den CWP als auch den aktuellen Memory Stack Pointer, ohne daß der Benutzer Kenntnis davon hat (mit Ausnahme von Traps).

Meistens benutzt ein Programm mehr Fenster, als ein Chip zur Verfügung stellen kann. Ein Overflow Trap zum Betriebssystem hin tritt auf, wenn alle Fenster vor einem save voll sind; ein Underflow Trap tritt auf, wenn alle Fenster leer vor einem restore sind. Der Overflow/Underflow-Handler legt die 16 Register eines Fensters im Speicher ab oder holt sie aus diesem hervor. Im Betriebssystem der Sun-Workstations (SunOS) werden diese Fenster im Memory Stack abgelegt.

Obwohl das Überlappen von entsprechenden ins und outs eine effiziente Möglichkeit bietet, den Parameterdurchsatz zu beschleunigen, ist der prinzipielle Vorteil im cache-ähnlichen Verhalten der Registerfenster zu sehen. Prozeduraufrufe und -rücksprünge werden über entsprechende Auf- und Abwärtsbewegungen im Execution Stack gesteuert, wobei allgemein diese Bewegungen ein gewisses Level im Stack nicht verlassen. Registerfenster sind dann effektiv, wenn die durchschnittliche Anzahl dieser Sprünge auf dem Stack kleiner ist als die Zahl der Fenster.

Registerfenster bieten gegenüber festen Registern einige Vorteile: Durch das cache-ähnliche Verhalten wird die Anzahl der load- und store-Instruktionen, die vom Compiler veranlaßt werden, reduziert, da die in Registern gehaltenen lokalen und die Rücksprungadressen zwischen Prozeduraufrufen nicht explizit in den Memory Stack gesichert beziehungsweise von ihm geladen werden müssen. Daraus ergibt sich eine Verringerung der Austauschrate Chip/Cache und Cache/Memory – das bedeutet: weniger loads und *stores* und effizienter Gebrauch des Daten-Cache. Das gibt Chips mit load- und store-Instruktionen, die in mehreren Zyklen ablaufen, den Vorzug, weil weniger dieser Instruktionen ausgeführt werden müssen; es erlaubt ebenso den Einsatz von Multiprozessoren mit Verringerung des Verkehrs auf dem Memory-Bus.

Sowohl in inkrementellen Compiler-Umgebungen wie LISP wie auch in objektorientierten Sprachen wie Smalltalk lassen sich mit der Technik der Fenstereinteilung gute Ergebnisse erzielen. Im letzteren Fall ist es unpraktikabel, mit IRA („Interprocedural Register Allocation") zu arbeiten, weil Prozedur Linking dynamisch zur Laufzeit geschieht. Allgemein betrachtet sind Registerfenster besser als eine Architektur, aufsetzend auf festen Registern mit IRA-basierenden Compilern und Bindern, weil Fenster sich dem Laufzeitverhalten

von Programmen ohne Probleme dynamisch anpassen können.

Interprocedural Register Allocation (IRA)

SPARC schließt IRA oder damit verbundene Optimierungen nicht aus, weil die call- und return-(jmpl-)Instruktionen sich von denjenigen unterscheiden, die den CWP erhöhen oder wieder zurücksetzen (save und restore). Ein Compiler und Binder kann für entsprechende Sprachen, aufsetzend auf der fensterorientierten Technik, IRA-basierende Optimierung durchführen, womit sogar die Anzahl der Window-Overflows und -Underflows sowie die load/store-Instruktionen reduziert werden.

Dynamische Trace-Daten von großen C-Programmen mit Compilern, die kein IRA oder darauf beruhende Optimierung durchführen, zeigen, daß 20 Prozent der ausgeführten SPARC-Instruktionen load- und store-Instruktionen sind, einschließlich der Behandlung von Underflows/ Overflows bei Fenstern. Bei RISC-Prozessoren ohne Registerfenster beträgt diese Anzahl 30 Prozent unter Verwendung von IRA, und 50 Prozent bei CISC-Prozessoren mit wenig (kleiner als 16) Registern, allerdings ohne IRA. Der Speicherverkehr durch load/store wird um etwa 30 Prozent reduziert. Weiterhin zeigen diese Trace-Daten, daß vier Prozent der save-Instruktionen auf die Window-Overflow-Handler zurückzuführen sind. Weil 16 Register bei einem Overflow-Trap gespeichert werden, ist dieser Vorgang gleichzusetzen mit zwei Drittel der Zeit des Schreibens eines Registers in den Speicher je save-Instruktion.

Overflow und Underflow bei Registerfenstern wird mit Hilfe eines Statusregisters der IU erkannt – dem Window Invalid Mask (WIM), für den Benutzer nicht zugänglich oder sichtbar. Das Betriebssystem benutzt das WIM zur Kennzeichnung von Fenstern oder Fenstergruppen, so daß beim Auftreten eines Overflow/Underflow-Traps eine save- oder restore-Instruktion den CWP auf ein gekennzeichnetes Fenster zeigen läßt. Um beim Auftreten eines Windows-Traps das übliche LIFO-Prinzip bei der Stack-Behandlung mit überlappenden Fenstern beibehalten zu können, setzt das Betriebssystem ein WIM-Bit zur

Literatur

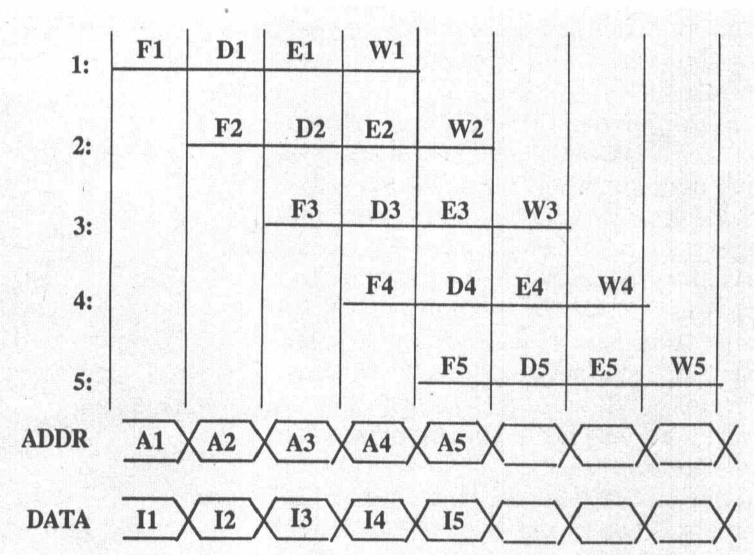
Robert B. Garner: „SPARC-Architecture". Sun Technology: Sommer 1988.
„SPARC Architecture Manual". Sun Microsystems Inc.
M. Namjoo, A. Agrawal: „Implementing SPARC". Sun Technology: Winter 1988.

eindeutigen Identifizierung der Grenze zwischen altem und neuem Fenster.

Prozeßauslagerung

Anzumerken ist hier, daß bei Prozeßauslagerung nur das Fenster mit den gültigen Daten in den Speicher gesichert wird. Durchschnittlich sind das die Hälfte der vorhandenen Fenster (abzüglich des reservierten Trap-Fensters): Auf einer Sun 4/260 mit 16,67 MHz unter SunOS beträgt diese Zeit zum Sichern der drei Fenster 15 Mikrosekunden. Die Registerfenster können vom Betriebssystem verwaltet werden, um eine sehr schnelle Trap-Behandlung zu erzielen. Beim Auftreten eines Traps oder Interrupts verhindert die IU weitere Traps und der CWP wird um eins verringert, womit das nächste Registerfenster für den Trap-Handler allokiert wird. Wenn nun der Trap-Handler das nächste Fenster in den Speicher sichert, bevor weitere Traps wiederzugelassen werden, oder wenn diese infolge der Kürze überhaupt nicht nötig ist, wird das Nichterkennen von Interrupts auf nur einige Zyklen begrenzt. (Dabei ist zu beachten, daß der Trap-Handler nur fünf der 24 Register des Trap-Fensters benutzen kann: Zwei der 8 locals werden von der IU-Trap-Hardware zum Sichern der Programm-Pointer benutzt, 1 local mit dem PSR belegt. Die outs und ins des neuen Fensters sind deshalb nicht verwendbar, weil die ins die momentanen outs derjenigen Prozedur sind, in der der Trap auftrat, und die outs gültige ins einer früheren Prozedur sein können, wenn alle Fenster benutzt sind.) Bei spezieller Hardware wie I/O-Controllern oder realzeitfähigen Systemen, die sehr hohe Anforderungen an den Informationsaustausch stellen, kann die Behandlung von Registerfenstern von dieser Standardmethode abweichen. Ein mögliches Schema dafür sieht eine Einteilung der Fenster zu Paaren vor, wobei ein Paar für einen Prozeß reserviert wird. Erreichen läßt sich das durch Markierung der Fenster im WIM; diese Markierung macht das Fenster für den jeweils anderen Prozeß unzugänglich und ermöglicht das Prozeßauslagern zwischen Fensterpaaren innerhalb weniger Zyklen.

Vier-Phasen-Instruktions-Pipeline



Zusätzlich zu den zu Fenstern zusammengefaßten Registern und zum WIM hat die IU noch andere Statusregister. Das PSR enthält ein Benutzer/Superuser-bit, einen MC-68000-kompatiblen Zustands-Code, das 4 bit lange Interrupt-Level des Prozessors (PIL), FPU- und CP-Zustands-Bit, den CWP- sowie eine 8 bit lange Versions-(Implementierungs-)Nummer. Das Trap-Basis-Register (TBR) enthält eine programmierbare Basisadresse für die Trap-Tabelle und ein 8 bit langes Feld, das den Typ des momentanen Traps identifiziert. Ähnlich dem WIM sind PSR und TBR nur über das Betriebssystem zu erreichen.

Architektur der Floating-Point-Unit

Die Fließkomma-Einheit (Floating-Point-Unit FPU) hat nur 32 bit lange Register: Doppelt genaue Werte belegen ein Einzelpaar, Werte mit höherer Genauigkeit eine ausgerichtete Vierergruppe von Registern. Die Floating-Point-Register können demnach maximal entweder 32 Werte mit einfacher Genauigkeit, 16 Werte mit doppelter Genauigkeit oder acht Werte mit höherer Genauigkeit belegen. Von außen wird auf die FPU-Register nur mit Speicherlade- und Speicherlese-Instruktionen zugegriffen. Es gibt keine direkte Verbindung zwischen IU und FPU. Programme, die doppelt genaue Zahlen verarbeiten, können durch „double instructions" für Floating-Point-Lade- und -Speichervorgänge in ihrer Laufzeit verbessert werden.

Obwohl die Benutzerebene der Floating-Point-Architektur dem ANSI/IEEE-754-1985-Standard entspricht, müssen einige leichte Abweichungen vom Standard, etwa gradueller Underflow, mit Software behandelt werden. Durch Erzeugen eines unfertigen oder nicht vorhandenen Floating-Point-Traps zeigt eine FPU, daß sie kein korrektes ANSI/ISEE-754-1985-kon-

formes Ergebnis liefern kann. Die fehlenden Funktionen werden dann durch System-Software simuliert. Falls die FPU nicht präsent ist, muß diese in ihrer Gesamtheit durch das Betriebssystem emuliert werden.

SPARC erlaubt arithmetische Floating-Point-Operationen wie Multiplizieren und Addieren, die gleichzeitig miteinander, aber auch mittels Integer-Instruktionen und Lade- und Speichervorgängen von Fließkommazahlen durchgeführt werden. Zum Beispiel ist es möglich, das Vorausladen eines Floating-Point-Wertes auszuführen, während eine Floating-Point-Multiplikation und -Addition läuft.

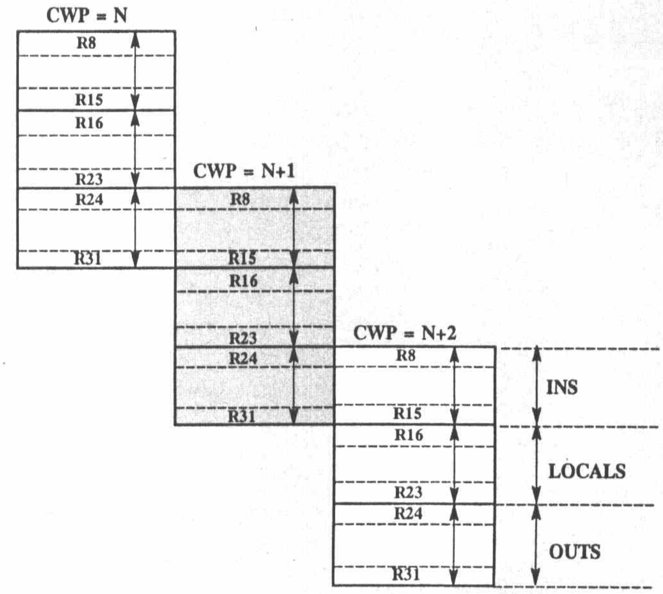
Der Grad der möglichen Zahl gleichzeitiger Aktionen ist implementierungsabhängig – die FPU kann beispielsweise mehrere Multiplizierer und Addierer enthalten. In allen aktuellen Konstruktionen werden arithmetische Floating-Point-Instruktionen unter Cache-Einsatz realisiert.

Die FPU führt erforderliche Registersperrungen durch, so daß keine neue Floating-Point-Instruktion erfolgen kann, bevor nicht alle Operanden berechnet worden sind. Die zur gleichen Zeit stattfindende Ausführung bleibt dem Benutzer verborgen. Die Programme bringen die gleichen Ergebnisse, einschließlich der Reihenfolge der Floating-Point-Traps, wie wenn die Instruktionen nacheinander abliefen. Die Registersperrungen können über eine sogenannte „Register-Kerbe" verwirklicht werden: diese besteht aus 1 bit je Register, das anzeigt, ob ein Register auf einen Ergebniswert wartet.

Wegen der vorher erwähnten Gleichzeitigkeit kann der IU-Programmzähler eine oder mehrere Instruktionen der Floating-Point-Instruktion vorziehen, die einen Floating-Point-Trap erzeugt hat. Im allgemeinen muß er deshalb nicht auf die Floating-Point-Instruktion zeigen, die den Trap erzeugt hat. Um Traps gut handhaben zu können, verfügt die FPU auch über eine Floating-Point-Queue. Die nach FIFO arbeitende Warteschlange enthält alle nicht erfüllten Floating-Point-Arithmetik-Instruktionen und ihre Speicheradresse zur Zeit des Traps. Die Länge dieser Floating-Point-Queue ist abhängig von der Mikroarchitektur der FPU (das Laden und Speichern von Floating-Point-Werten wird nicht in die Queue aufgenommen). Oben in der Schlange steht die noch nicht beendete Instruktion, die den Trap hervorgerufen hat. Die anderen Instruktionen in der Schlange sind noch nicht ausgeführt. In SunOS werden alle Instruktionen, die zum Zeitpunkt des Auftretens des Traps in der Schlange stehen, mit Software emuliert.

Das vom Benutzer erreichbare Floating-Point-Status-Register (FSR) enthält Statusinformationen: im einzelnen sind dies Kontroll-Bits, die Traps ermöglichen, Bits zur Indizierung von Ausnahmen und Ausnahmestatus-Bits für die fünf möglichen Trap-Typen. Da ungenaue Traps auftreten können, muß das Bit „inexact-trap-enable" in allen SPARC-Chips verfügbar sein. Im FSR befinden sich weiterhin der Floating-Point-Zustands-Code, der Floating-Point-Type und das „Floating-Point queue not empty"-Bit. (Nach dem Auftreten von Floating-Point-Traps bekommt das Betriebssystem solange Input von FQ, bis diese leer ist.)

SPARC-Register-Modell



Koprozessor-Architektur

SPARC unterstützt einen von der jeweiligen Umsetzung her definierten Koprozessor. Wie die FPU hat auch der Koprozessor eigene Register, führt Instruktionen mittels IU und FPU gleichzeitig aus und kann durch ein Zustands-Bit im PSR inaktiv gesetzt werden. Ähnlich der FPU, die exaktes Trapping unterstützt, enthält der CP eine Koprozessor-Queue, die alle noch nicht zu Ende geführten Koprozessor-Instruktionen und die Adressen zum Zeitpunkt des Absprungs enthält.

Der RISC-Prozessor MC 88100

David Jones

Der Autor
David Jones studierte 1978 bis 1982 Electrical and Electronic Engineering an der Heriot-Watt University. Danach war er bei Motorola Ltd. in verschiedenen Positionen tätig. Seit Juni 1988 ist er Senior-High-End/ASIC-FAE in München.

Mit Einführung der Familie der 88000er Prozessoren ist eine neue 32-bit-RISC-Architektur entstanden. Dieser Artikel vermittelt einen ersten Überblick über diese innovative Prozessorarchitektur.

Es liegt in der CISC-Struktur selbst begründet, daß weiter steigende Verarbeitungsleistungen mit dieser Technologie schwer zu erreichen sind. Deshalb haben Firmen wie AMD, Sun und Acorn Architekturen eingeführt, die die erwünschte Leistungssteigerung bieten – in Form der RISC-(Reduced Instruction Set Computer-)Prozessoren.

Motorola bietet einen RISC-Prozessor an, der mit dem 1,5-µm-HCMOS-Prozeß hergestellt wird, und zwar den 88100 sowie eine Cache- und Speicherverwaltungs-Einheit (CMMU), den 88200. Der Prozessor vereint zahlreiche RISC-Eigenschaften ebenso wie eine Reihe von CISC-Prinzipien, was zu einer Leistung von etwa 17 MIPS bei einer Taktfrequenz von 20 MHz führt. Erste Labormuster der beiden Bausteine lagen 1987 vor; Muster werden gegenwärtig sowohl bei Motorola als auch bei Anwendern getestet. Einige wenige Konstruktionsprinzipien bilden die wesentliche Grundlage des RISC-Mikroprozessors, nämlich

- Ausführung sämtlicher Befehle in einem einzigen Taktzyklus;
- keine Microcode-Logik, alle Befehle sind hardwaremäßig vorgegeben;
- weniger und einfachere Befehle als bei den CISC-Rechnern;
- einfache Adressierungsarten;
- speziell ausgerichtete Architektur, um höhere Computersprachen zu nutzen.

Um Befehle in nur einem Taktzyklus ausführen zu können, wurde eine Architektur im „Harvard"-Stil mit getrennten Code- und Datenbussen gewählt. Das macht es möglich, Befehle aus dem Code-Speicher beziehungsweise -Cache getrennt von Informationen aus dem Daten-Speicher beziehungsweise -Cache abzurufen. Der 30-bit-Adreßbus bedient einen 4-GByte-Adreßumfang. Alle Befehlswörter haben 32 bit. Ein Synchronbus mit zwei Taktzyklen, der im Pipeline-Verfahren arbeitet, ermöglicht den Zugriff auf Befehle und Daten in nur einem Taktzyklus.

Niedriger Befehlssatz

Der 88100 enthält lediglich 51 Befehle, einschließlich des vollen Fließkomma-Satzes nach IEEE 754. Mit Ausnahme von Speicherzugriffs-, ganzzahligen Multiplizier- und Dividier- sowie Fließkomma-Befehlen und einigen wenigen Ablaufsteuer-Befehlen werden alle Befehle in einem Taktzyklus ausgeführt. Es gibt weder einen Mikrocode noch Erweiterungswörter, die zur Befehlsausführung verwendet würden. Jeder Befehl ist hardwaremäßig in der Logik vorhanden, um so die Decodierzeit zu verringern. Mit Hilfe von einfachen Adressierungsarten wird auch die effektive Adreßbestimmungszeit gesenkt.

Um Ausgleich für die bei manchen Befehlen erforderlichen zusätzlichen Taktzyklen zu schaffen, werden Speicherzugriffs- und Fließkomma-Befehle von speziell hierfür bestimmten Ausführungs-Einheiten abgearbeitet. So wird vermieden, daß für den Prozessor bei mehrzyklischen Befehlen Wartezeiten entstehen.

Das Befehlsformat ist so beschaffen, daß jedes Bitfeld für Register oder Adressierungsarten etc. sich bei allen Befehlsgruppen an der gleichen Stelle im Wort befindet und so die Decodierlogik vereinfacht wird.

RISC-Erweiterungen

Die Architektur des 88100 erweitert die RISC-Philosophie mit dem Ziel der Leistungsoptimierung und verwirklicht insbesondere eine sorgsam abgestimmte Parallelität. Bei vorhandenen CISC-Architekturen werden schon in großem Umfang Pipelines verwendet. Ihre Länge schwankt zwischen drei und acht Stufen. Jedoch wurde bisher in der Regel nur eine Pipeline benutzt, und zwar bei Abruf-, Decodier- und Ausführungs-Befehlen. Wenngleich hierdurch eine beträchtliche Leistungssteigerung erreicht wird, so wirkt sich dies im Falle eines Sprungbefehls doch nachteilig aus.

Der 88100 enthält demgegenüber fünf befehlsausführende Einheiten (Execution Units) mit zugehörigen Pipelines. Diese Ausführungs-Einheiten bewirken weitgehende Parallelität, indem sie den parallelen Ablauf von fünf Operationen im Pipeline-Verfahren ermöglichen. So kann der Prozessor beispielsweise gleichzeitig mit dem Zugriff auf einen Programmspeicher einen Integer-Befehl ausführen, auf einen Datenspeicher zugreifen, einen Fließkomma-Befehl durchführen sowie einen Fließkomma- oder Ganzzahl-Multiplizier-Befehl ausführen.

Zusätzlich machen es die Pipelines für jede der obengenannten Ausführungs-Einheiten möglich, daß bis zu fünf Fließkomma-Befehle gleichzeitig bearbeitet werden, während noch sechs Fließkomma- und/oder Ganzzahl-Multiplizier-Befehle ablaufen, außerdem bis zu drei Datenspeicher-Zugriffe und bis zu drei Befehlsabrufe.

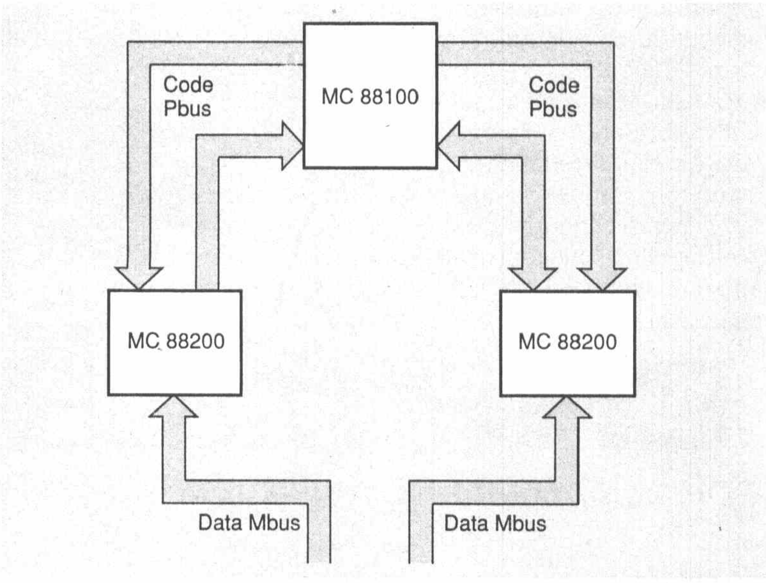
Eine weitere Funktion, der sogenannte „verzögerte Sprung", kann in den Pipelines vorbereitet werden. Das ermöglicht es, daß zunächst der nach dem Sprungbefehl abgerufene Befehl ausgeführt wird, bevor der Sprung stattfindet. Dieser nach dem Sprungbefehl abgerufene Befehl wird ausgeführt, während der zum Sprung gehörige Zielbefehl vom Speicher abgerufen wird; auf diese Weise wird eine Pipeline-Verzögerung ausgeschlossen.

Die 88100-Architektur stellt einen Rechner mit Parallelbetrieb dar, wobei alle Ausführungs-Einheiten Parallelität dort verwirklichen, wo dies auf die einzelnen Operationen anwendbar ist. So werden beispielsweise Befehlscodierung und Abruf eines Quellenoperanden gleichzeitig von derselben Ausführungs-Einheit durchgeführt. Befehlscodierung und Zieladreß-Bestimmung werden parallel mit sequentieller Adressengenerierung ausgeführt. Um diese Steigerung im internen Busverkehr zu ermöglichen, werden drei interne Datenbanken eingesetzt. Sie bewirken ein eher gleichzeitiges als sequentielles Abrufen von Operanden. Hierdurch wird ebenfalls paralleles Lesen und Beschreiben von Registern unterstützt, so daß synchron innerhalb eines Taktzyklus zwei Quellen- und ein Zielregister angesprochen werden können.

Zusätzliche Erweiterungen der grundlegenden RISC-Architektur umfassen Bitfeld-Befehle für Grafik- und Kontrollanwendungen. Ein Speicher-Austauschbefehl wurde in der Weise verwirklicht, daß der Bus während des Austauschprozesses blockiert ist, wodurch Semaphor-Register in Multiprozessor-Systemen ohne Behinderung durch andere Prozessoren aktualisiert werden können.

In der Architektur ist auch eine Fehlertoleranz-Anordnung vorgesehen: Zwei Prozessoren sind durch eine ODER-Verknüpfung miteinander verbunden; einer wird zum Master bestimmt, der andere über die Aussage des MST-Pins zum Kontrollprozessor (Checker). Alle Ausgänge

Fehlererkennung im Shadow-Verfahren



des Checkers mit Ausnahme des Fehler-Pins ERR sind dann nicht beschaltet und werden als Eingänge überwacht. Der Checker vergleicht seine internen Ergebnisse mit denjenigen, die von den floatenden Pins abgelesen werden. Stimmt ein Ergebnis nicht überein, steuert der Chekker seinen ERR-Pin an. Zusätzlich überprüft der Master ebenfalls, ob der interne Wert der von ihm gelieferten Signale dem der externen gleicht. Wenn ein Fehler auftritt, etwa ein externer Kurzschluß, steuert der Masterprozessor den ERR-Pin an. Die externe Logik kann dann auf dieses Fehlersignal hin in Aktion treten und den Fehler mitteilen oder den fehlerhaften Prozessor-Knotenpunkt umgehen.

Architektur

Die interne Architektur des 88100 setzt sich aus fünf unabhängigen Ausführungs-Einheiten zusammen. Die Mehrzahl der Befehle ist registerbezogen und funktioniert in einem dreigliedrigen System. Das bedeutet, es gibt zwei unabhängige Quellen und ein separates Zielregister für das Ergebnis, damit sichergestellt wird, daß es nicht zu einer Zerstörung von Daten kommt. Zur Unterstützung dieser Operation innerhalb eines Taktzyklus werden drei parallele 32-bit-Busse für die Quelle eins, die Quelle zwei und für das Zielregister eingesetzt.

Diese drei Busse sind dafür verantwortlich, daß Informationen intern jeweils dann zu den jeweiligen Funktionseinheiten geleitet werden, wenn sie dort benötigt werden. Eine Ablaufsteuerung (Sequencer) kontrolliert jegliche interne Busaktivität zu den verschiedenen Einheiten in Abhängigkeit von den Ergebnissen der Befehlsausführung. Alle Register sind 32 bit breit und nicht zweckgebunden. Zwei Register werden jedoch vom Prozessor intern für spezielle Funktionen verwendet.

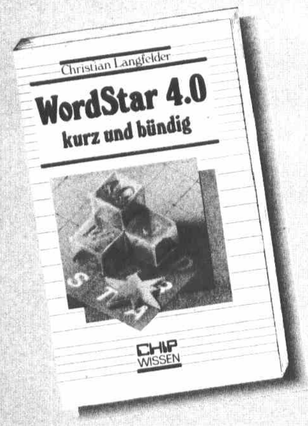
Daten werden aus einem externen Speicher oder zu ihm hin oder auch über die CMMU in eine Pipeline in der Dateneinheit gebracht, die unabhängig von den anderen Funktionen des Prozessors arbeiten kann. Daraus folgt, daß Daten bereits abgerufen werden können, bevor der Prozessor danach verlangt oder daß Daten in die Pipeline statt direkt in den Speicher geschrieben werden können, um so die Verzögerung um eine zusätzliche Taktperiode durch die Ausführung des Speicherbefehls auszugleichen. Eine ähnliche Pipeline ist für den Befehlsaufruf eingerichtet, der durch die unabhängige Befehlseinheit abgewickelt wird.

Ganzzahlige Rechenoperationen werden über eine zweckgebundene Integer- und eine Fließkomma-Einheit durchgeführt. Hier ermöglicht es eine Pipeline in der Fließkomma-Einheit, daß Mehrfachtakt-Befehle in die gesamte Taktsteuerung der Befehlsausführung des Prozessors integriert werden. Dies bewirkt, daß ein Mehrfachtakt-Befehl so in Erscheinung tritt, als liefe er in der kürzestmöglichen Zeit ab.

Die Fließkomma-Einheit weist zwei Pipelines auf: Addier-, Subtrahier-, Dividier- und Konvertierungsbefehle werden mittels der einen Pipeline ausgeführt, während Multiplizierbefehle in der anderen abgewickelt werden. Addier-, Subtrahier- und Konvertierungsbefehle mit ein-

CHIP WISSEN

Die kompetente Buchreihe rund um den PC



Christian Langfelder:

WordStar 4.0 kurz und bündig

184 Seiten, 46 Bilder, Hardcover
35,– DM/ISBN 3-8023-**0219**-2

Dieses Buch verfolgt einerseits das bewährte Konzept seines Vorgängers „WordStar kurz und bündig" und macht den Leser andererseits mit den neuen Eigenschaften und Möglichkeiten von WordStar 4.0 bzw. den Zusatzprogrammen von WordStar 4.0 Extra vertraut. Es ist somit für Einsteiger in der Textverarbeitung als auch für Umsteiger auf die neue Version bestens geeignet.



Hans Mahnke:

Strukturiert programmieren in FORTRAN 77

mit Microsoft FORTRAN 4.0
128 Seiten, 20 Bilder, Hardcover
30,– DM/ISBN 3-8023-**0221**-4

Dieses Buch wendet sich an alle, die die Programmiersprache FORTRAN erlernen wollen, also Studenten, Techniker und PC-Benutzer, die gut strukturierte Programme brauchen.
Durch seinen einfachen Aufbau und die Kombination von Programmieren mit der Sprache FORTRAN eignet es sich besonders gut zum Einsatz an Schulen, bei Lehrgängen und Seminaren sowie als Schulungsunterlage in Firmen, wenn dort FORTRAN in Großrechenanlagen programmiert ist.
Aus dem Inhalt:
- ★ Aufbau von FORTRAN-Programmen
- ★ COMMON-Bereiche
- ★ BLOCK-DATA-Unterprogramme
- ★ EQUIVALENCE-Anweisung
- ★ Ein- und Ausgaben

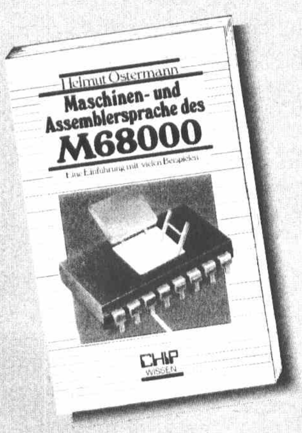


Fred Wagenknecht:

Das 68000er-Animationsbuch

Bewegte Grafik auf Amiga und Atari ST
284 Seiten, 134 Bilder
99 Listings, Hardcover
48,– DM/ISBN 3-8023-**0211**-4

Dieses Buch für Grafikfreaks bietet Designern und Grafikinteressierten einen unkomplizierten Einstieg in die Computeranimation. Anhand relativ einfacher BASIC-Programme wird der Leser Schritt für Schritt in spannende und schöpferische Grafikprogramme eingeführt. Diese 2D- und 3D-Experimente sind überschaubar strukturiert und ausführlich dokumentiert. Der Entwurf der Objekte und der Bewegungsabläufe sind durchschaubar und führen zu sofortigen Erfolgserlebnissen.



Helmut Ostermann:

Maschinen- und Assemblersprache des M68000

Eine Einführung mit vielen Beispielen für Amiga, NDR-Klein-Computer und Sinclair QL
368 Seiten, 62 Bilder
zahlr. Listings, Hardcover
48,– DM/ISBN 3-8023-**0174**-9

Angesprochen sind Leser mit Vorkenntnissen in der Programmierung eines Mikroprozessors. Während Anfänger mit Hilfe des Glossars und eines Anhangs den Einstieg finden. Die vielen Beispiele wurden auf Amiga, Sinclair QL und NDR-Klein-Computern entwickelt.
Seine erstaunliche Leistungsfähgkeit und die klare Struktur der Adressierungsarten ermöglichen erfolgreiches Programmieren.
Der Mikroprozessor 68000 von Motorola hat durch Computer wie Amiga, Macintosh, Atari St und Sinclair QL in kurzer Zeit eine weite Verbreitung gefunden.

Haben Sie schon den neuen „CHIP- Katalog"?
Bestellen Sie gleich!

VOGEL Buchverlag Würzburg
Postfach 67 40 · 8700 Würzburg 1

facher Genauigkeit erfordern fünf Taktzyklen, während Multiplizierbefehle sechs benötigen. Aus diesem Grund sind je eine 5-Stufen- und eine 6-Stufen-Pipeline in dieser Ausführungs-Einheit vorhanden. Dividierbefehle werden ausgeführt, indem sie für jedes erforderliche Genauigkeitsbit eine Stufe der Pipeline durchlaufen.

Die Tatsache, daß die Fließkomma-Einheit auf dem Chip integriert ist, ermöglicht den Daten- und Befehls-Einheiten freien Zugriff auf die externen Busse ohne Behinderung durch eine CPU, die gerade

0	h	l	l	h	g	l	l	g	n	e	0
	s	o	s	i	e	t	e	t	e	q	

eq Equal
ne Not equal
gt Greater than
le Less than or equal
lt Less than
ge Greater than or equal
hi Higher than
ls Lower than or same
lo Lower than
hs Higher than or same

0	o	i	i	o	g	l	l	g	n	e	n	c
	b	n	b	u	e	t	e	t	c	q	c	p

cp comparable
nc uncomparable
eq Equal
ne Not equal
gt Greater than
le Less than or equal
lt Less than
ge Greater than or equal
out Out of range
ib In range or on boundary
in In range
ob Out of range or on boundary

Als Alternative zu einem Condition-Code-Register werden Prädikate in einem Zielregister aus den frei verfügbaren Registern ausgewertet. Die Gruppe von Werten auf der linken Seite ist für die Adressierung mit 16-bit-Feld bestimmt. Die Gruppe rechts bezieht sich auf die dreigliedrige Registerbefehl-Adressierung.

ein externes Busprotokoll gegenüber einem externen Koprozessor durchführt; auf diese Weise entsteht kein Verlust an Übertragungsleistung.

Es ist die Integer-Einheit, die die meisten Befehle des 88100 ausführt und zusätzlich alle diejenigen Operationen, die für die Speicheradressierung und die Programmabfolge erforderlich sind. Diese Einheit wurde deshalb in drei Abschnitte unterteilt. Für Ganzzahl-Arithmetik ist eine vollständige 32-bit-ALU vorhanden, die den gesamten Bereich der arithmetischen und logischen Standardoperationen abdeckt, wie es bei den meisten CISC-Mikroprozessoren auftritt. Sprungbefehle werden in einer Sprungeinheit ausgeführt; sie ist für die Erzeugung und Speicherung der Rückkehradresse zuständig und sorgt für die Sprungverzögerung in den Pipelines. Alle Bitfeld-Befehle werden in einer getrennten Bitfeld-Einheit ausgeführt. Die Trennung dieser drei Aufgaben ergibt erhebliche Parallellität in der Befehlsausführung. So könnten drei dieser Befehlstypen gleichzeitig mit verschiedenen Registern ausgeführt werden.

Dateneinheit

Ausgerichtete 32-bit-Datenzugriffe werden über die Dateneinheit gesteuert. Hierbei handelt es sich ebenfalls um eine separate Ausführungs-Einheit, die eine 30-bit-Arithmetik-Einheit für die Adreßbestimmung enthält. Eine dreistufige Pipeline beschleunigt die Operation des Datenabrufs durch diese Einheit. Die erste Stufe der Pipeline definiert die erforderliche Adresse; zusätzlich werden bei einem Speicherbefehl Daten für das interne Registerarchiv abgerufen. Die zweite Stufe enthält den Treiber für den externen Bus und – bei einem Speicherbefehl – ebenfalls den Treiber für den Datenbus. In der dritten Stufe wird die Antwort vom externen Speichersystem beziehungsweise der CMMU überwacht. Für eine Speicheroperation wird der Datenbus gelesen, für eine Schreiboperation wird ein Register im Registerarchiv geladen.

Befehlseinheit

Die Befehlseinheit ist ebenfalls für eine ganze Reihe von Aktivitäten zuständig. Die Befehlseinheit ruft Befehle vom Port ab, führt jedoch zusätzlich bereits die ersten Schritte der Befehlsdecodierung aus und ermöglicht es so, daß Instruktionen innerhalb eines Taktes von der Befehls- zur Integer-Einheit übermittelt werden, wo dies möglich ist. Um das verwirklichen zu können, enthält auch die Befehlseinheit intern eine dreistufige Pipeline.

Innerhalb dieser Pipeline befinden sich drei Befehlszeiger. Der Ausführungsbefehl-Zeiger (Execution Instruction Pointer – XIP) enthält die Adresse des aktuellen Befehls, der gerade in der Integer- beziehungsweise Fließkomma-Einheit ausgeführt wird. Der zweite ist der Folgebefehl-Zeiger (Next Instruction Pointer – NIP), derjenige Befehl, der gerade vom Speicher oder der CMMU empfangen wird. Da Daten bei dem auf die Adresse folgenden Takt empfangen werden, weist der Abrufbefehl-Zeiger (Fetch Instruction Pointer – FIP) auf den Speicherplatz des nächsten Befehls, auf den der Zugriff erfolgen soll. Zusätzlich zu diesen zweckgebundenen Einheiten erlaubt die Architektur des Prozessors bis zu acht Sonderfunktions-Einheiten (Special Function Units – SFU). Dazu gehört die Fließkomma-Einheit nach IEEE-Spezifikation, wie sie auf dem 88100 als erstem Mitglied der Prozessor-Familie vorhanden ist.

Der Zugriff auf alle diese Sonderfunktions-Einheiten erfolgt über die normalen Befehlsabfolgen. Der einzige Unterschied bei den entsprechenden Befehlen besteht in der Decodierung der SFU-Nummer in den höheren Bits des 32-bit-Befehlswortes. Die Fließkomma-Einheit ist auf SFU = 1 voreingestellt.

Der 88100 arbeitet entweder im User- oder im Supervisor-Modus – wobei typischerweise bei Applikationen und entsprechender Software im User- und beim Betrieb von System-Kernprogrammen im Supervisor-Bereich gearbeitet wird.

Ablaufsteuerung

Sämtliche Ablaufsteuerungs-Vorgänge innerhalb des 88100 werden über den Sequencer reguliert. Diese Ausführungs-Einheit ist für den Registerzugriff, die Behandlung von Exceptions, die Zuteilung der internen Busbenutzung und die Erzeugung von Steuersignalen für die Integer- und Sonderfunktionen zuständig.

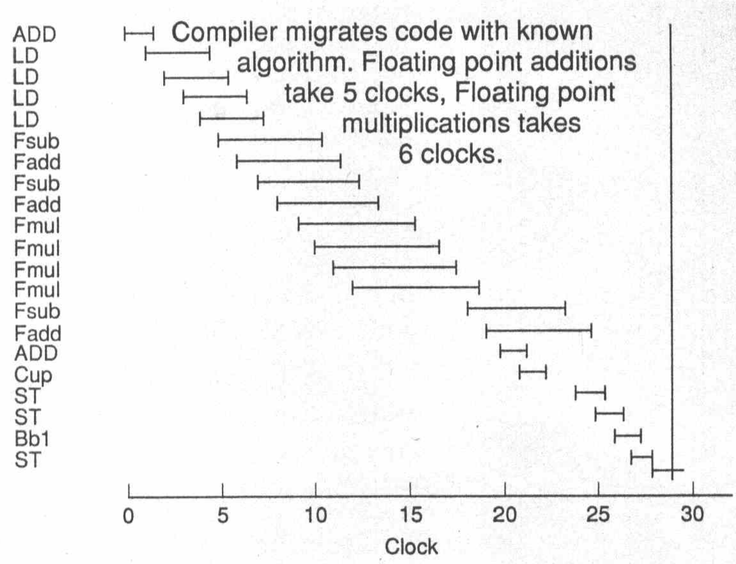
Eng mit der Funktion des Sequencers verbunden ist die Registerbenutzung. Da es sich um einen Parallelprozessor handelt, ist es nicht immer einfach festzulegen, welche der Register tatsächlich zur gleichen Zeit benutzt werden. Um diese Situation in den Griff zu bekommen, verfügt jedes frei verfügbare Register über ein zugehöriges „Anzeigentafel"-(Scoreboard-)Bit, eine Art Markierung für die Registerbenutzung. Wenn die Integer- oder die Fließkomma-Einheit gerade ein Register benutzt, wird ein Bit im Scoreboard-Register gesetzt. Hierdurch ist sichergestellt, daß bei paralleler Ausführung von Befehlen nicht ein Quellenoperanden-Register frequentiert wird, das ein vorangegangener Befehl noch ausgerechnet hat. Deshalb ist der Prozessor nicht darauf beschränkt, jeden Befehl in Reihe zur Vollständigkeit auszuführen.

Exceptions

Die Zuständigkeit des Sequencers für die Exception-Behandlung ist ebenfalls von vorrangiger Bedeutung für die Leistungsfähigkeit des Systems. Zwei Arten von Exceptions können in der CPU auftreten: bestimmte und unbestimmte. Eine bestimmte Exception („Ausnahme") ist diejenige, bei der der genaue Prozessorkontext und die exakte Ursache bekannt sind, beispielsweise Interrupts, Busfehler etc. Bei einer bestimmten Exception verfügt der Prozessor über alle zur Wiederherstellung erforderlichen Informationen.

Unbestimmte Exceptions sind solche, wie sie beispielsweise durch den Überlauf einer Fließkomma-Einheit eintreten. In diesem Fall wird nicht der gesamte Kontext zur Wiederherstellung benötigt, solange die gerade in Durchführung befindliche Operation bekannt ist.

Für einen maximalen Datendurchsatz ist es wesentlich, daß der Sequencer die Fähigkeit besitzt, auf alle anstehenden bestimmten Exceptions hin tätig zu werden, bevor unbestimmte Exceptions behandelt werden.



Der dem Compiler zugeführte optimierte Code führt die 22 Befehle in nur 29 Takten mit einer Leistung von 6,9 MFlops bei einer Taktfrequenz von 20 MHz aus. Der Compiler bewegt den Code entsprechend bekannten Algorithmen. Fließkomma-Additionen benötigen fünf, Fließkomma-Multiplikationen sechs Takte.

Registerarchiv

Sämtliche Ausführungs-Einheiten werden durch diejenigen Befehle miteinander verbunden, die in den 31 frei verfügbaren, im Registerarchiv enthaltenen Registern ablaufen. Jedes Register besitzt ein zugehöriges Scoreboard-Bit. Das Registerarchiv weist drei Ports auf, zwei Ausgänge und einen Eingang von den internen Bussen.

Eine Methode, die Informationszuführung zu beschleunigen (Feed Forwarding), wurde bei den internen Bussen verwirklicht. Hierbei wird das Ergebnis für den Bestimmungsort aus der laufenden Operation auf einen der Quellenbusse gesetzt, der für die Ausführung des nachfolgenden Befehls benötigt wird.

Dadurch ist gewährleistet, daß die Einheit, die auf die Information wartet, diese zum frühestmöglichen Zeitpunkt erhält. Das trifft auch dann zu, wenn beide Quellenbusse gerade belegt sind. In die-

sem Fall werden beide Quellenoperanden mit Hilfe des Feed Forwarding vom Zielbus empfangen.

Innerhalb des Registerarchivs befindet sich eine autonome Zuteilungseinrichtung, die das Register-Beschreibungsprotokoll steuert. Dieser Mechanismus dient dazu, den Schreiboperationen in die Register durch die Ausführungs-Einheiten Prioritäten zuzuordnen. Die Integer-Einheit kommt hier an erster Stelle, gefolgt von der Fließkomma- und schließlich der Dateneinheit.

Programmiermodell

Die interne Architektur des 88100-Prozessors besteht aus 32 frei verfügbaren Registern und 64 Steuerregistern. Beim Einschalten arbeitet der Prozessor in der Supervisor-Betriebsart und bietet Zugriff auf alle Steuerregister für die Prozessor-Initialisierung und auf sämtliche frei verfügbaren Register. Typischerweise arbeitet jede System-Software in dieser Betriebsart. In der normalen User-Betriebsart stehen dem Benutzer alle nicht zweckgebundenen Register r0 bis r31 zur Verfügung sowie zusätzlich die Statusregister der Fließkomma-Steuerung.

Anders als bei r0 und r1 sind diesen Registern keine speziellen Aufgaben zugewiesen wie zum Beispiel Adreß-, Stack-Pointer-, Index- oder Datenregister. Somit ist Flexibilität bei der Programmierung möglich. Immer wenn ein Zugriff auf das Register r0 erfolgt, gibt es den Wert 0 zurück. Dies kann dazu eingesetzt werden, den Inhalt eines beliebigen Registers auf effiziente Weise zu löschen. Zusätzlich wird r1 dazu benutzt, den Wert der Rückkehradresse eines Unterprogramms aufzunehmen, da ja kein Stack vorhanden ist.

Auf den ersten Blick scheint es unmöglich, mit diesem Prozessor zu arbeiten, denn es gibt weder im User- noch im Supervisor-Modus einen Stack-Pointer. Noch schlimmer – es gibt auch kein Condition-Code-Register. In vielerlei Hinsicht ist es auf diese Funktionen zurückzuführen, daß – bei sehr hohem Datenduchsatz – in konventionellen Architekturen Verzögerungen in der Verarbeitung auftreten. Offensichtlich muß es also eine andere Methode zur Behandlung von Exceptions, von Sprung- und Vergleichsbefehlen geben, wenn kein Stack- und kein Condition-Code-Register vorhanden sind.

In die Steuerregister der Integer-Einheit sind Shadow- und Exception-Time-Register integriert. „Schattenbildung" ist eine Technik zur Aufzeichnung der internen Pipeline-Register während der Befehlsausführung. Die in den Shadow-Registern enthaltenen Informationen sind die drei Befehlsanzeiger (Pointer) und das Scoreboard-Register. Für die Zeit einer Ausnahmebehandlung werden diese Registerwerte „eingefroren" und somit eine Kopie des Prozessorkontextes zum Zeitpunkt des Auftretens der Exception angelegt. Die Exception-Time-Register werden dazu benutzt, andere interne Register des Prozessors für die Zeit einer Exception aufzunehmen und so die vollständige Wiederherstellung bei einer Störung sicherzustellen.

Aufgrund der Parallel-Architektur des 88100 würde der Einbau eines Condition-Code-Registers – beispielsweise zur Auswertung der Ergebnisse eines Vergleichsbefehls – die Notwendigkeit mit sich bringen, daß ein Befehl Priorität über einen anderen erhielte, um Zugriff auf das Condition-Code-Register erlangen zu können. Als Folge davon müßte der zweite Befehl so lange warten, bis der erste Befehl unter dieser (diesen) Bedingung(en) ausgeführt ist. Bei einer RISC-Architektur ist dies nicht praktikabel, deshalb wurde auch kein Condition-Code-Register eingerichtet. Statt dessen verwendet der 88100 eine Kette von Prädikaten, die in allen frei verfügbaren Registern ausgewertet wird. Beim Vergleichsbefehl werden zwei Quellen-Opcodes miteinander verglichen, das Ergebnis dieses Vergleichs wird dann in einem Zielregister abgelegt.

Befehle werden mit Hilfe von drei Adressier-Betriebsarten abgerufen. Die erste ist die Befehlsregister-Adressierung mit 9-bit-Vektortabellen-Index. Hier können bis zu vier mögliche Bedingungen mit einem Quellenregister geprüft werden. Wenn die Bedingung gültig ist, wird der spezifizierte 9-bit-Vektor-Offset für den Trap-Befehl nach Bitprüfung zu dem Vektor-Basisregister addiert, um so den FIP zu ergeben. In der zweiten Adressierungsart wird die Befehlsregister-Adressierung mit 16-bit-Feld für Sprung- und Trap-Befehle benutzt. Schließlich werden Sprünge über einen 26-bit-Wort-Offset ausgeführt. Dabei wird dieser Wert zum FIP addiert, um den FIP zu erzeugen.

Um Datendurchsätzen mit hoher Geschwindigkeit Rechnung zu tragen, können Busse so konfiguriert werden, daß sie im Pipeline-Verfahren arbeiten.

Versteckte Intelligenz in industriellen Anwendungen

Alan Steinberg

Bestimmt ist jedem klar, der sich auch nur ein bißchen in der Elektronikindustrie auskennt, daß diese relativ junge Branche gegenwärtig stürmisch vorandrängt. Sichtbare Auswirkungen hiervon sind elektronische Maschinen, sei es im industriellen oder im persönlichen Bereich, die immer intelligenter werden. Und mit der Konstruktion dieser fortlaufend anspruchsvolleren, intelligenteren Maschinen müssen auch die elektronischen Bauteile mithalten. Eine Schlüsselrolle spielen hierbei spezielle Mikroprozessoren, genannt Embedded Controller („eingebettete Controller").

Die industriellen und technischen Steuerungen wurden lange von Single-Chip-Mikrocomputern oder „Mikro-Controllern" beherrscht, als die sie nach der Einführung des Mikroprozessors bekannt wurden. Diese Chips bewährten sich auf dem Gebiet der industriellen Steuerungen, ihnen wurde aber niemals die Aufmerksamkeit zuteil, die der Mikroprozessor im Computer genießt. Dieser Bausteintyp wurde vor allem dadurch bekannt, daß er seinen Platz in den neuesten und auch den stärksten Rechnern fand.

Mikrocontroller, die in Wirklichkeit viel verbreiteter sind als Mikroprozessoren, hatten die weniger publikumswirksame Aufgabe, solchen Maschinen versteckte Intelligenz für schematisch auszuführende Arbeiten mitzugeben. Außerdem konnten sie typischerweise nach der Programmierung durch den Hersteller der Maschine vom Anwender nicht mehr umprogrammiert werden. Im allgemeinen setzten sie sich statt dessen im Umfeld der industriellen Steuerung durch.

Neuentwicklungen

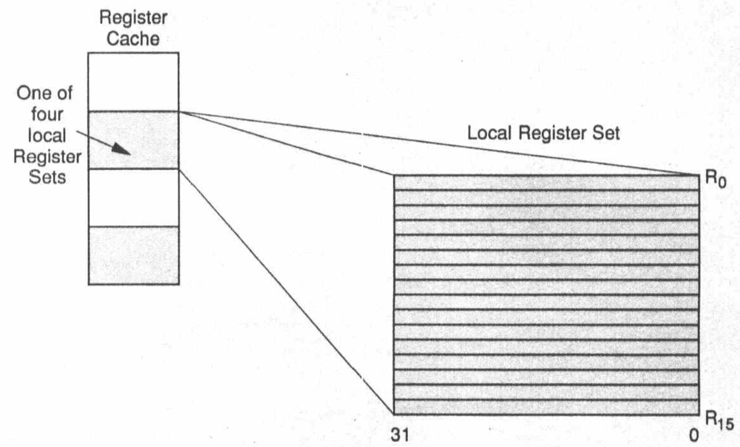
Angesichts der neueren Entwicklungstendenzen im industriellen Steuerungsbereich war Intel zu der Überzeugung gelangt, daß weder seine traditionelle Mikrocontroller noch seine programmierbaren Mikroprozessoren den steigenden Geschwindigkeitsanforderungen im industriellen Bereich genügen könnten. Vielmehr glaubte das Unternehmen, daß hierfür neue Bausteine entwickelt werden müßten. Und um den steigenden Anforderungen im 32-bit-Markt zu genügen, entwickelte Intel daher zwei Strategien zur Befriedigung dieser Bedürfnisse im industriellen Bereich.

Intels 80376 für industrielle Steuerungen

Einer von Intels neuen industriellen Controllern wurde aus dem bekannten 32-bit-Mikroprozessor 80386 entwickelt. Er kann zusammen mit einem hochintegrierten Koprozessor für Ein- und Ausgabe (I/O) betrieben werden, der DMA-Kanäle und einen Interrupt-Controller für den Einsatz unter Echtzeitbedingungen enthält. Der 80376 kann aber auch alleine – ohne den I/O-Koprozessor 80370 – arbeiten, sofern oben erwähnte I/O-Fähigkeiten nicht benötigt werden. Als Chip-Satz bilden der 80376/80370 einen praktikablen Aufwärtspfad für alle auf einem 16-bit-Prozessor 80186 oder 80C186 basierenden industriellen Entwicklungen nach 32 bit. Durch die Kompatibilität des 80376 zum 80186 können Anwender ihre Software einfach übertragen. Der hochintegrierte Peripheriebaustein 80370 sorgt für

Alan Steinberg ist bei der Intel Corporation in Chandler/Arizona als Marketing Manager Embedded Controller tätig

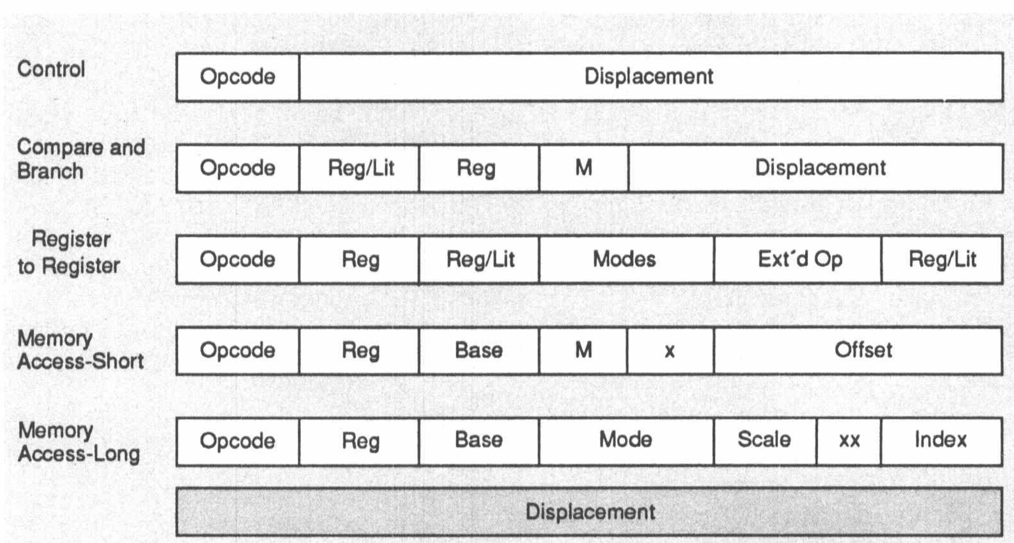
Mehrere Registersätze sind auf dem Chip gespeichert



eine effiziente Ausnutzung der vorhandenen Platinenfläche, da er mehrere periphere Bausteine ersetzt. Er enthält einen 32-bit-DMA-Controller mit acht Kanälen, drei Interrupt-Controller, vier programmierbare 16-bit- und einen dynamischen Refresh-Controller. Der 80376 hat eine interne 32-bit-Architektur, einen externen 16-bit-Daten-Bus und kann 16-MByte-Arbeitsspeicher direkt (physikalisch) ansprechen. Damit erreicht er rund 70 Prozent der Leistung eines 80386. Intel hat hierbei einige Features des 80386 weggelassen, die typischerweise in industriellen Anwendungen nicht gebraucht werden, wie beispielsweise das Paging oder 16-bit-Operationen im Real-Mode. Verglichen mit einer VAX 11/780 von Digital Equipment (DEC) erreicht der 80376 2,5 bis 3 Millionen Instruktionen pro Sekunde (MIPS).

Für diesen neuen industriellen Prozessor ist ein kompletter Satz von Entwicklungshilfsmitteln wie In-Circuit-Emulatoren, Debugger und anderen Tools erhältlich. Zur Software-Entwicklung kann jeder Personal-Computer mit einem 80386 als Prozessor benutzt werden.

Die Instruktionsformate



Der Markt der Embedded Controller

Im Laufe der Jahre hat man festgestellt, daß industrielle Anwender sich wenig darum kümmern, ob es sich bei einer CPU um einen Mikroprozessor oder einen Mikrocontroller handelt und ob sie von einem Standard-Betriebssystem unterstützt wird. Viel wichtiger ist in diesem Bereich, daß die benötigten Funktionen kostengünstig zu realisieren sind, daß der Prozessor einfach in ein System zu integrieren ist und daß er auch in der nächsten Prozessorgeneration noch genug leistet. Und dabei ist auch nicht zu vergessen, daß sich die Software-Investition beim Übergang zu einem neuen Mitglied der Familie möglichst vollständig erhalten lassen sollte.

Industrielle Controller werden zur Steuerung in Systemen, zur Erfassung von Daten und in ereignisgesteuerten Anwendungen (zum Beispiel Motorsteuerungen) eingesetzt.

Neue Prozessorarchitektur

Vor kurzem hat Intel nun eine gänzlich neue Mikroprozessor-Konstruktion vorgestellt, den 80960 – übrigens Intels erste neue Prozessorarchitektur seit zehn Jahren. Mit einer neuen 32-bit-Architektur orientiert sich dieser Prozessor speziell an den Anforderungen des industriellen Marktes und enthält alle Möglichkeiten, die von einem Embedded Controller erwartet werden.

Die Bauweise 80960 ist total verschieden von der bekannten 32-bit-Konstruktion des 80386. Während der 80386 für Anwendungen in programmierbaren Rechnern wie Personal-Computern oder Workstations ausgelegt ist, wurde die Architektur des 80960 speziell für den Bereich der Embedded Controller geschaffen. Die Bauweise dieses Prozessors verbindet Konstruktionsmerkmale der RISC-Technik mit Eigenschaften für industrielle Anwendungen, um eine einfache und kostengünstige Entwicklung zu ermöglichen. Im Gegensatz zu reinen RISC-Maschinen, die ausschließlich in einer Hochsprache programmiert werden, läßt

sich der 80960 in Assembler oder einer Hochsprache programmieren.

Im Unterschied zu reinen RISC-Maschinen, die ihre theoretische Arbeitsgeschwindigkeit nur bei Verwendung von sehr gut optimierenden Compilern tatsächlich erreichen, erzielt die 80960-Familie ihre vollen Leistungen dank ihres vielseitigeren Instruktionssatzes auch ohne einen speziell optimierenden Compiler.

Ein 80960-Prozessor verwaltet zum Beispiel seine eigene Befehls-Pipeline inklusive „Silicon Locking". Während bei normalen RISC-Prozessoren eine Programmierung in Assembler fatale Folgen haben kann, ist dies beim 80960 nicht der Fall. Es empfiehlt sich vielmehr sogar, spezielle I/O-Routinen in Assembler zu schreiben. Die RISC-Merkmale des 80960 werden nur bei komplexen Vorgängen in einer „Embedded"-Umgebung ausgenutzt, um die integrierten Schaltungen kostengünstig und leicht programmierbar zu erhalten.

Ein großer Teil der 80960-Konstruktion ist dafür ausgelegt, den Rechen- und Datendurchsatz des Prozessors durch erhöhte Parallelisierung zu verbessern. Hierfür benutzt die CPU ein effizientes LOAD/STORE-Speicherzugriffs-Modell, in dem Code und Daten in einem Cache zwischengespeichert werden. Hierbei werden auch Maschinenbefehle, von denen viele Ein-Takt- oder Zwei-Takt-Befehle sind, überlappend ausgeführt; der 32 bit breite Adreß/Daten-Bus ist gemultiplext.

In dieser innovativen „Embedded"-Architektur hat Intel drei neue Produkte vorgestellt. Auch spätere Nachfolgeprodukte werden sich dieser Bauweise bedienen, dabei aber spezielle neue Funktionen entsprechend den Bedürfnissen der industriellen Anwender bieten. Der modulare Kern des 80960 wird es in Zukunft möglich machen, die CPU in eine komplette Familie anwendungsspezifischer Produkte zu übernehmen, beginnend mit applikationsspezifischen Standard-Bausteinen (ASSP) bis hin zu „Core based" ASICs.

Die ersten Produkte der 80960-Familie – der 80960KB, der 80960KA und der 80960MC – sind im wesentlichen für das größte Segment im industriellen Bereich ausgelegt, den Bereich der Systemsteuerungen. Der 80960KB enthält eine integrierte Fließkomma-Recheneinheit und ist derzeit schon lieferbar. Der 80960KA, der in Kürze erhältlich sein wird, enthält diese Recheneinheit nicht. Der 80960MC ist eine spezielle militärische Version des Prozessors mit Fließkomma-Einheit für den Einsatz im gesamten Bereich der Luft- und Raumfahrt. Die militärische Version enthält weiterhin eine MMU (Memory Managing Unit – Speicherverwaltungseinheit) und ermöglicht zusammen mit dem Bus-Erweiterungsprozessor (BXU) 82965 den Aufbau von speziellen fehlertoleranten Systemen. Die 80960K-Prozessoren sind für alle Anwendungen bestimmt, bei denen es auf hohe Geschwindigkeit unter Echtzeitbedingungen ankommt, wie etwa numerische Berechnungen, Bildverarbeitung, Laserprinter, Roboter, industrielle Prozessoren oder den Bereich der Telekommunikation.

Die Architektur des 80960K

Genau wie andere RISC-Prozessoren erreichen die 80960K-Chips ein hohes Arbeitstempo, indem sie externe Speicherzugriffe durch ihre „LOAD/STORE-Architektur" minimieren. Bei ihr wird der Arbeitsspeicher nur für LOAD- und STORE-Befehle angesprochen. Alle anderen Befehle (arithmetische, logische, vergleichende, verzweigende, Bit-Operationen etc.) laufen zwischen den Registern des Bausteins ab. Die geringe Zahl der Speicherzugriffe steigert die Systemgeschwindigkeit nicht unerheblich. Diese spezielle Arbeitsweise ermöglicht es dem Anwender, sein System basierend auf langsameren und damit billigeren Speicherbausteinen (zum Beispiel dynamische RAMs) zu entwickeln.

Um die für externe Speicherzugriffe benötigte Zeit noch mehr einzuschränken, enthält die 80960K-Architektur zwei Mechanismen, um Code und Daten auf dem Chip zwischenzuspeichern: einen Instruktions-Cache und mehrere Sätze von lokalen Registern. Der 512 KByte große Zwischenspeicher für Befehle sorgt dafür, daß die Befehls-Pipeline der CPU mit einem stetigen Strom von Instruktionen versorgt wird und iterative Programmteile wie Programmschleifen komplett auf dem Chip ausgeführt werden können.

Das vereinfachte Befehlsformat schlägt sich in einheitlich 32 bit (vier Byte) langen Instruktionen nieder, die immer auf passenden Wortgrenzen liegen müssen (Alignment). Dieses Alignment erübrigt das Ausrichten der Befehle auf gerade Wortgrenzen in der Instruktions-Pipeline, wie es normalerweise erforderlich ist.

Der 80960KB kann in der maximal möglichen Geschwindigkeit mit der Au-

ßenwelt verkehren, da sein Daten-Bus an die interne Cache-Struktur angepaßt wurde. Die „Line Size" des Cache stimmt genau mit der maximalen Burst-Länge der Instruktionszyklen überein. Das heißt, vier aufeinanderfolgende Befehlswörter können im sogenannten Burst-Modus zusammen geladen und direkt im Cache abgelegt werden. Diese Arbeitsweise wurde zusammen mit den Instruktions-Cache entwickelt, damit der Prozessor auch bei höchster Geschwindigkeit noch mit relativ langsamen Arbeitsspeichern auskommt.

Eine weitere Steigerung der Systemgeschwindigkeit wird durch überlappende Ausführung einzelner Befehle erreicht. Ein LOAD-Befehl läßt zu, daß noch beliebige andere Instruktionen ausgeführt werden, solange er seinen Speicherzugriff nicht vollständig ausgeführt hat. Die Bauweise des 80960KA/KB ermöglicht diese Arbeitsweise mit der für RISC-Prozessoren charakteristischen Technik des „Register Scoreboarding".

Eine gemeinsame Charakteristik von RISC-Prozessoren ist, daß normale Befehle wie MOVE, ADD, SUB, logische oder Verzweigungs-Operationen in einer minimalen Anzahl von Taktzyklen (möglichst in einem) ausgeführt werden können. Da diese einfachen Funktionen in den meisten Programmen am häufigsten auftreten, wird hierdurch die Systemgeschwindigkeit erheblich gesteigert. Die 80960-Konstruktion unterstützt dieses Konzept sowohl durch sein LOAD/STORE-Speicherzugriffs-Modell als auch durch die Elimination der Wortanpassung in der Instruktionswarteschlange. Das ergibt in über 50 Prozent aller Fälle Befehle, die in einem einzigen Maschinenzyklus ausgeführt werden.

Der Prozessor 80960KB ist mit vielen Registern ausgestattet: Er hat sechzehn 32 bit breite globale Register, sechzehn lokale Register und vier 80 bit breite Register für Fließkommazahlen. Weitere Merkmale umfassen einen 512-Instructions-Cache von 512 bit, einen Register-Cache mit vier Sätzen lokaler Register, vier Gigabyte linearen (nicht segmentierten) Adreßraum, eine zur Norm IEEE 754-kompatible Fließkomma-Einheit und einen Interrupt-Controller mit 32 Interrupt-Ebenen und 256 Interrupt-Vektoren, eine wertvolle Erweiterung für Interrupt-gesteuerte Echtzeit-Anwendungen.

Der gemultiplexte 32-bit-Bus des Prozessors erlaubt mit 16 Byte pro Übertragung im Burst-Modus eine Gesamt-Datenrate von 53,3 MByte/s. Ein mit 20 MHz laufender 80960KB erreicht so 20 Burst-MIPS und 7,5 MIPS im Normalbetrieb. Die Fließkomma-Einheit auf dem Chip leistet vier Millionen Whetstones pro Sekunde.

Verfügbarkeit und Entwicklungsunterstützung

Der 80960KB und die 80960KA-Version ohne Fließkomma-Einheit sind bereits in Stückzahlen erhältlich. Den 80960MC kann man ebenfalls bereits erhalten; er ist aber entsprechend seiner Zuverlässigkeit nach Militärnorm wesentlich teurer.

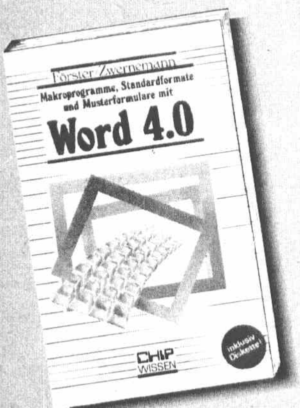
Die Hilfsmittel zur Hardware- und Software-Entwicklung für den 80960KB bestehen aus einem kompletten Satz von Werkzeugen, darunter auch Produkte von unabhängigen Herstellern, mit denen Intel Verträge zur vollen Unterstützung dieser neuen Chip-Konstruktion getroffen hat. Ein Evaluations-Board von Intel, genannt EVA-960KB, ist zum Kennenlernen des Prozessors und für Benchmarks gedacht. Das EVA-960KB ermöglicht es Programmierern, sofort mit der Entwicklung von Programmen zu beginnen und sie zu testen.

Der ASM-960KIB, aus Dienstprogrammen, Linker/Loader, Makro-Präprozessor und Archivierung bestehender Kit, ist dazu bestimmt, den Entwicklungsprozeß zu vereinfachen. Ein optimierender C-Compiler mit ANSI-Erweiterungen erzeugt Quellcode und Symbole für den Assembler. Dieser Compiler mit Namen iC-940KB unterstützt auch „Inline"-Assembler-Code. Ferner ist Intel dabei, weitere Entwicklungs-Tools bereitzustellen.

Impressum

CHIP PLUS erscheint monatlich als Beilage zu CHIP. Anschrift der Redaktion: Schillerstr. 23a, 8000 München 2, Tel. 0 89-5 14 93-0, Verlag: Vogel-Verlag KG, Max-Planck-Str. 7/9, 8700 Würzburg, Tel. 09 31/4 18-0. Redaktionsdirektor: Richard Kerler. Redaktion: Martin Stübs (verantw.), Ulrich Parthier. Grafische Gestaltung: Hans Kuh, Gabi Raffius. Autoren dieser Ausgabe: Georg Pöll, Uri Mayer, Ulrich Parthier, David Jones, Alan Steinberg. Verlagsdirektor: Dr. Andreas Kaiser. Anzeigen: Friedrich Mangold. Satz, Litho, Druck, Verarbeitung, Versand: Vogel-Druck Würzburg. Der Verlag haftet nicht für unverlangt eingesandte Manuskripte und Fotos.

CHIP WISSEN



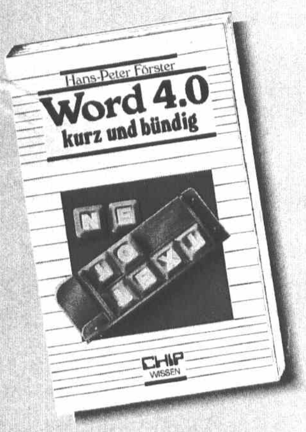
Hans-Peter Förster/Martin Zwernemann:

Makroprogramme, Standardformate und Musterformulare mit Word 4.0

ca. 152 Seiten, 42 Bilder, Hardcover
48,– DM (mit 5,25″-Diskette)
ISBN 3-8023-**0246**-X

Jeder der mit Word 4.0 Standardformulare ausfüllt, Serientexte schreibt und Etiketten bedruckt oder eigene Makroprogramme und Druckformatvorlagen erstellen möchte, braucht dieses Buch mit Diskette.

Auf der Diskette befinden sich nahezu alle Etikettenformate, Formate für Endlospapiere und Standardformulare. Die Makroprogrammierung, das Arbeiten mit Druckformaten und Serientexten wird Schritt für Schritt anhand von Praxis-Beispielen erklärt.



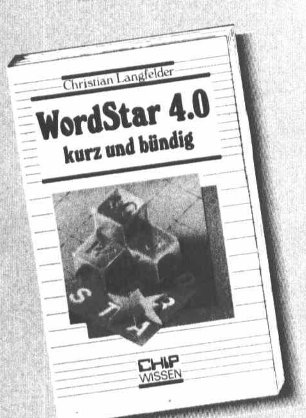
Hans-Peter Förster/Martin Zwernemann:

Word 4.0 kurz und bündig

256 Seiten, 67 Bilder, Hardcover
43,– DM/ISBN 3-8023-**0215**-X

Dieses Arbeitsbuch richtet sich an interessierte Einsteiger sowie an Word-4.0-Anwender, die schnell den Umgang mit dem komfortablen Textverarbeitungsprogramm erlernen wollen. Es ist so aufbereitet, daß es auch als Begleitmaterial für Word-4.0-Schulungen geeignet ist. Aber auch zum Selbststudium ist es in Ergänzung zum Word-4.0-Handbuch ein Hilfsmittel, das viele Tips und Tricks verrät.

* Wichtige Grundfunktionen
* Texte erstellen, korrigieren, speichern
* Texte laden, korrigieren, gestalten
* Texte direkt/indirekt formatieren
* Ausdruck
* Textbausteine
* Rechtschreibprogramm u.a.

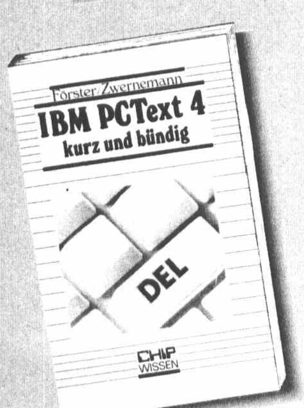


Christian Langfelder:

WordStar 4.0 kurz und bündig

184 Seiten, 46 Bilder, Hardcover
35,– DM/ISBN 3-8023-**0219**-2

Dieses Buch verfolgt einerseits das bewährte Konzept seines Vorgängers „WordStar kurz und bündig" und macht den Leser andererseits mit den neuen Eigenschaften und Möglichkeiten von WordStar 4.0 bzw. den Zusatzprogrammen von WordStar 4.0 Extra vertraut. Es ist somit für Einsteiger in der Textverarbeitung als auch für Umsteiger auf die neue Version bestens geeignet.



Hans-Peter Förster/Martin Zwernemann:

IBM PCText 4 kurz und bündig

ca. 200 Seiten, 47 Bilder, Hardcover
38,– DM/ISBN 3-8023-**0247**-8

Ziel dieses Buches ist es, dem Anwender des IBM Textverarbeitungsprogrammes PCText 4 einen Ratgeber zur Seite zu stellen, der alle Funktionen und Befehle knapp aber präzise vorstellt.

Immer wieder wird der Leser beim Durcharbeiten des Buches zur direkten Eingabe der notwendigen Tasten aufgefordert. Die Knappheit des Textes und die Fülle der Informationen machen das Buch zu einem wichtigen Nachschlagewerk bei der täglichen Arbeit und damit zu einer praktischen Ergänzung zum IBM-Handbuch.

Haben Sie schon den neuen „CHIP- Katalog"?
Bestellen Sie gleich!

VOGEL Buchverlag Würzburg
Postfach 6740 · 8700 Würzburg 1

CHIP WISSEN

Die kompetente Buchreihe rund um den PC



Hans-Walter Beilstein:

Wie man in C programmiert

Eine Einführung anhand von Turbo C und DeSmet-C
504 Seiten, 143 Bilder, Hardcover
53,– DM/ISBN 3-8023-**0189**-7

Dieses Buch führt gründlich und umfassend in die Besonderheiten von C ein. C-Kenntnisse werden nicht vorausgesetzt, es ist aber von Vorteil, wenn der Leser sich schon mit einer anderen Programmiersprache beschäftigt hat. Zunächst erfährt er alles über C (u.a. C-Befehle und C-Funktionen, Ein- und Ausgaberoutinen, Variablen, C-Vektoren, C-Bibliothek), dann werden C-Tools wie Präsentationsgrafik oder Windowtechnik angewendet und in vorhandene Programme eingebaut. Aufgaben mit Lösungen zu jedem Kapitel helfen, die erworbenen Kenntnisse anzuwenden und zu überprüfen.

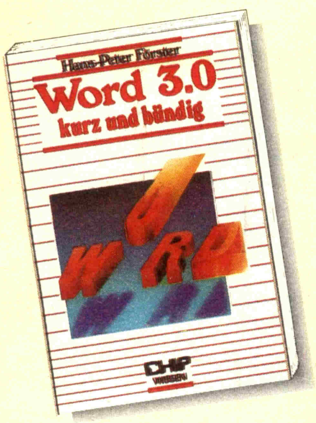


Hans-Peter Förster/Martin Zwernemann:

MultiMate 3.6 kurz und bündig

144 Seiten, 13 Bilder, Hardcover
33,– DM/ISBN 3-8023-**0208**-7

MultiMate ist ein komfortables, leicht bedienbares Textprogramm. Seine Kombination aus Menüsteuerung und Funktionstastenprinzip ermöglicht eine ganz auf das persönliche Vorgehen zugeschnittene Arbeitsweise. Das Buch beschreibt in kompakter Form die Funktionen und den Umgang mit MultiMate. Es richtet sich an interessierte Einsteiger, die schnell den Umgang mit diesem Programm beherrschen wollen. Alle MultiMate-Befehle stehen griffbereit zur Verfügung. Das Buch ist somit eine nützliche Ergänzung zu den Handbüchern.



Hans-Peter Förster/Martin Zwernemann:

Word 3.0 kurz und bündig

Texte erfassen, gestalten, drucken
200 Seiten, 61 Bilder, Hardcover
38,– DM/ISBN 3-8023-**0188**-9

Dieses Buch erklärt die Funktionen der deutschen Version 3.0 des Textprogramms MS-Word. Es hilft dem Anwender, alle Word-Befehle auf kompaktem Raum stets griffbereit zu haben. Word 3.0 steuert Laserdrucker und Satzmaschinen direkt an. So lassen sich sehr flexibel Druckwerke produzieren. Zusammen mit einer Datenbank und einer Desktop-Software sind die Anwendungsmöglichkeiten äußerst vielfältig. In kurzer verständlicher Form sprechen die Autoren auch diese Punkte an und stellen anhand praktischer Beispiele interessante Anwendungen vor.



Rainer Schwinn:

dBase III PLUS kurz und bündig

260 Seiten, 26 Bilder, Hardcover
38,– DM/ISBN 3-8023-**0180**-3

dBase III PLUS unterscheidet sich von seinem Vorgänger dBase III durch einen erweiterten Leistungsumfang, z.B. seine Netzwerkfähigkeit. Der Leser wird anhand leicht nachvollziehbarer Beispiele in das Arbeiten mit der deutschen Version dieser Datenbank eingewiesen. Dabei lernt er u.a. wie er

- die Benutzerführung durch einen Assistenten in Anspruch nimmt und das System um Hilfe fragt
- Dateistrukturen erstellt, verändert und kopiert
- Daten in Dateistrukturen eingibt und modifiziert
- individuelle Bildschirmmasken erzeugt
- Dateien sortiert und indiziert
- wie er mit einem Programmgenerator ohne Programmierkenntnisse eigene Programme erstellt.

Haben Sie schon den neuen „CHIP-Katalog"?
Bestellen Sie gleich!

VOGEL Buchverlag Würzburg
Postfach 6740 · 8700 Würzburg 1